Vierte Dimension

2+3/2010

Das Forth–Magazin

für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten

In dieser Ausgabe:

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D–22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH–Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmeßtechnik und bauen z. Z. eine eigene Produktpalette auf.

Know–How Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential–Dosierwaagen, DMS–Messverstärker, 68000 und 68HC11 Prozessoren, Sigma–Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX–Verleih

Seit unserem Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX–Komponenten, dass ich meine privat eingebrachten Dinge nun Anderen, vorzugsweise Mitgliedern der Forth–Gesellschaft e. V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO–RCX–Set, so wie es für ca. 230,– € im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO Steine.

Anfragen bitte an
Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist „narrensicher“!

RetroForth
Linux · Windows · Native
Generic · L4Ka::Pistachio · Dex4u
Public Domain
`http://www.retroforth.org`
`http://retro.tunes.org`

Diese Anzeige wird gesponsort von:
EDV–Beratung Schmiedl, Am Bräuweiher 4, 93499 Zandt

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690–380
Fax: 02461/690–387 oder –100
Karl–Heinz–Beckurts–Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug– und Sondermaschinen, Fuzzy Logic.

FORTecH Software GmbH
Entwicklungsbüro Dr.–Ing. Egmont Woitzel

Bergstraße 10	D–18057 Rostock
Tel.: +49 381 496800-0	Fax: +49 381 496800-29

PC–basierte Forth–Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software–/Systementwurf. Mehr als 15 Jahre Erfahrung.

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

Ingenieurbüro
Klaus Kohl–Schöpe

Tel.: 07044/908789
Buchenweg 11
D–71299 Wimsheim

FORTH–Software (volksFORTH, KKFORTH und viele PDVersionen). FORTH–Hardware (z.B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs– und Meßtechnik.

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth–Gesellschaft e. V.
Postfach 32 01 24
68273 Mannheim
Tel: ++49(0)6239 9201-85, Fax: -86
E–Mail: `Secretary@forth-ev.de`
`Direktorium@forth-ev.de`
Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

Bernd Paysan, Ulrich Hoffmann
E–Mail: `4d@forth-ev.de`

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

Januar, April, Juli, Oktober jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Liebe Leser,

willkommen zur Doppelausgabe 2/3 des 26. Jahrgangs unseres Forth–Magazins.

Sicherlich werdet Ihr Euch fragen, wieso Ihr so lange kein neues Heft in Eurem Briefkasten gefunden habt und wieso Ihr nun ein Doppelheft bekommt. Nun – 2010 ist ein sehr trubeliges Jahr und es passieren sehr viele Dinge. Auch solche, die die Aufmerksamkeit weg von der Produktion der Vierten Dimension lenken. Die Redaktion ist mit nur zwei Mitgliedern schwach besetzt und wir könnten gut Hilfe brauchen, um Engpässe besser abfedern zu können.

Aber nicht nur an Verstärkung in der Redaktion mangelt es. Ein Blick in das Inhaltsverzeichnis verrät aber auch, dass der Kreis der Autoren immer noch die üblichen Verdächtigen umfasst und es offenbar für viele Leser eher interessant ist, zu konsumieren, als selbst über ihre eigenen Forth–Projekte zu berichten. Dabei lebt ein Vereinsmagazin wie die Vierte Dimension ja vom Zuspruch und der Mitarbeit der Mitglieder.

Nun - was hat der treue Autorenkern für das Doppelheft auf die Beine gestellt? Wir finden dort praktische Forth–Anwendungen im Hausautomations–Bereich, wie der Bericht über den *Desulfator* von Rolf Schöne und die *Solaranlagen–Steuerung* von Michael Kalus. Andere Artikel (wieder Michael Kalus) beschäftigen sich mit der Basistechnik des Parsens von *fremden Datenformaten* und der *genauen Taktgenerierung*. Außerdem erfahren wir von Fred Behringer im siebten Teil seiner Bootmanager–Artikelserie, wie man einen *Disketten–Bootmanager* in Forth programmiert. Martin Bitter berichtet uns über seine Erfahrungen mit *Minos, dem Rahmenwerk für graphische Benutzeroberflächen von BigForth* (und Vfx). *Vergesst* `forget` *nicht!* ruft uns wiederum Fred Behringer in seinem Artikel über temporäre anonyme Wort–Definitionen zu. Auf der Jahrestagung 2010 in Rostock ging der kleine *WikiReader*, den wir auch auf der Titelseite sehen, schon durch die Reihen. Carsten Strotmann fasst noch einmal für alle Leser zusammen, wo sich Forth in diesem Handgerät findet. Über jüngste Bestrebungen, auf den AVRs *mehr Platz für amforth* zu gewinnen, schreibt der amforth–Erfinder Matthias Trute.

Ende September fand in Hamburg die 26. EuroForth–Konferenz statt. Wir hatten zahlreiche Teilnehmer aus dem europäischen und außereuropäischen Ausland (England, Österreich, USA, Südafrika) aber nur einige aus Deutschland selbst. Das finde ich schade, denn es sind durchaus sehr spannende Themen, wie die Forth-CPU *J1* (entwickelt von James Bowman) oder das Distributions–Netzwerk *The Forth Net* (initiiert von Gerald Wodni), diskutiert worden. Ich kann mir nur schwer vorstellen, dass die deutschen Forther da nichts beizutragen gehabt hätten. Im Vorfeld der Konferenz fand, fast schon traditionell, das Forth-200x–Standardisierungs–Treffen statt. Die Arbeiten am so genannten *Snapshot*, ein runder Zwischenstand, der in Kürze veröffentlicht und dann wiederum auch über ANSI standardisiert werden soll, neigen sich dem Ende zu.

So — nun viel Spaß beim Lesen dieser Doppelausgabe. Schreibt mehr Forth–Artikel und schickt sie bitte an die Redaktion (vd@forth-ev.de)

Bitte meldet Euch auch, wenn Ihr Lust habt, beim Erstellen der Vierten Dimension mitzuwirken.

Ulrich Hoffmann

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können sie auch von der Web–Seite des Vereins herunterladen.
`http://www.forth-ev.de/filemgmt/index.php`

Die Forth–Gesellschaft e. V. wird durch ihr Direktorium vertreten:

Ulrich Hoffmann Kontakt: `Direktorium@Forth-ev.de`
Bernd Paysan
Ewald Rieger

arrayForth™

`comp.lang.forth`–Leser wissen es schon, allen anderen sei es nun verraten: Das Bündel aus dem neusten colorForth und den damit erzeugten Werkzeugen zum Umgang mit den Green–Array–Chips nennen sich nun *arrayForth*. Das CAD–System und die Chipdefinitionen selbst sind jedoch nicht enthalten, obwohl auch ganz in colorForth verfasst. Das colorForth und das arrayForth sind lebendige Entwicklungen und in andauerndem Fluss, werden entsprechend den Bedürfnissen der Chipentwickler weiter entwickelt. Es ist in nichts Standard, alles ist völlig eigenständig. Einfachheit und Brauchbarkeit sind deren einzige Leitlinien. Zukünftige Kompatibilität ist nicht garantiert. Es sei *mehr wie Ton in unseren Händen, nicht Granit.* Doch ist arrayForth der zugrunde liegende Softwarekörper für den Umgang mit diesen Chips, und daher nun für den freien Gebrauch veröffentlich.

Die Green–Array–Chips:

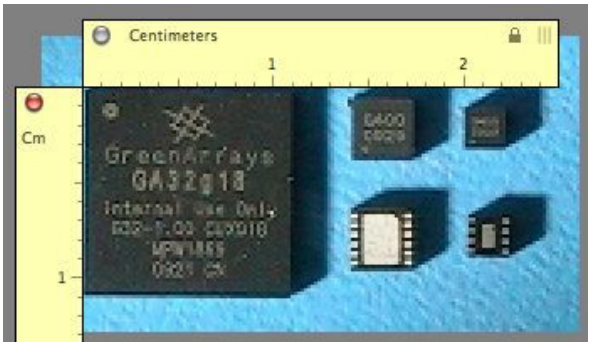
GA4 Ein 4–Computer–Chip in einem 8–Pin–Gehäuse. Der Chip misst 1x1 mm und wurde im 180nm–Prozess gefertigt.

GA32 Ein 32–Computer–Chip in einem 88–Pin–Gehäuse. Der Chip misst 2,5x2,5 mm und wurde ebenfalls im 180nm–Prozess gefertigt.

GA40 Ein 40–Computer–Chip. Der Chip wurde im 130nm–Prozess gefertigt.

GA144 Der bisher stärkste Chip enthält 144 g18–Computer, macht damit bis 100 Milliarden Operationen in der Sekunde. Dieser Chip wurde nun als Produktinsvorstufe hergestellt. Die Teste mit der Studie seien befriedigend verlaufen.

Link: `http://www.greenarraychips.com`



Nachricht aus Holland

Frans van der Markt schreibt uns:

> Am 14. August 2010 hat Leon Konings eine Vorführung über ArrayForth veranstaltet. ArrayForth läuft auf den neuen Chips von Green Arrays, Inc. So heißt der neue Betrieb von Chuck Moore und Freunden. Die Vorführung hat die ColorForth-artige Sprache ArrayForth und den Textverarbeiter behandelt. Es ist ein einfaches parallel arbeitendes Programm gezeigt worden. Dieses Programm läuft auf dem Simulator für den Chip GA144. Besondere Aufmerksamkeit ist auf den nicht einfachen Textverarbeiter und den Simulator gerichtet worden. Der GA144 ist noch nicht in Produktion gegangen. Es soll ein Chip mit einer Matrix von 8x18 Prozessoren werden. Der bisher stärkste Chip von Chuck Moore. Sehr bald wird auch ein ausführlicher Artikel auf unserer Website erscheinen. Da wird es über die Installation der GA144-Entwicklungs-Umgebung, des Textverarbeiters und des Simulators gehen. Es ist auch ein Kursus vorgesehen, in welchem die Herstellung eines GA144-Programms im Textverarbeiter dargestellt wird. Zur Zeit enthält es viele nützliche Links.

(fb)

amforth 3.9

Auch für Arduino und Atmega2561.

In der *amforth-devel* mailingliste hat Matthias Trute am 25.05.2010 mitgeteilt, dass nun im amforth repository ab Version 3.9 auch eine Anpassung des amforth-Kerns für den *Arduino* und Atmega2561 bereit liegt. Damit wurden die Liste der inzwischen vom amforth unterstützten Atmel 8-Bit-Prozessoren der atmega Familie zu den größeren Prozessoren hin ergänzt. Die Portierung auf diese Plattform samt der Atmega2561–Unterstützung war schwierig zu bewerkstelligen. Doch Andy Kirby hat dies dankenswerter Weise gut gelöst. Die Dateien in der sourceforge Sammlung (sourceforge file section) enthalten auch pre-compilierte Hex-Files für den *forthduino*.

Links

`mailto://amforth-devel@lists.sourceforge.net`
`https://lists.sourceforge.net/lists/listinfo/amforth-devel`
`http://amforth.sourceforge.net/`

(mk)

Neues vom Feigenblättchen

Von der Vijgeblaadje-Redaktion unserer niederländischen Forth-Nachbarn erreicht uns die folgende Meldung an die HCC-Fg-Mitglieder:

Aus Ermangelung an Artikeln war es uns diesmal nicht möglich, ein Vijgeblaadje anzufertigen und zu versenden. Vijgeblaadje 80, das jetzt in Bearbeitung ist, wird wahrscheinlich die letzte Ausgabe auf Papier bleiben. Sparmaßnahmen innerhalb der HCC lassen es auch nicht mehr verantworten, alle zwei Monate allen Mitgliedern ein Vijgeblaadje-Exemplar per Post zu schicken. In Zukunft werden wir euch in E-Mail-Form auf dem Laufenden halten. Die vorliegende Mail ist ein Beispiel dafür und sie wird an alle Mitglieder der Forth-Interessengemeinschaft geschickt, deren E-Mail-Adresse der Redaktion bekannt ist. Anmerkungen, Tipps, Artikel usw. bitte an den Unterzeichneten.

Frans van der Markt, Sekretariat Forth IG, f.l.van.der.markt at kader.hcc.nl

(fb)

Siehe auch unsere Zusammenfassung des Vijgeblaadje 79 auf Seite 9.

WikiReader — und wie weiter?

Carsten Strotmann hat uns darauf aufmerksam gemacht (siehe vorliegendes VD–Heft). Danke, Carsten. Innerhalb eines einzigen Tages habe ich das Ding von der Firma Pulster (Hinweis Carsten) per Nachnahme bekommen: 120,-- Euro. Das ist mir die Aussicht, mich wieder mal bastlerisch betätigen zu können, wert. Aber wie geht es weiter? Googlen ist nicht immer effizient: Wenn man eine Idee hat und wirklich etwas sucht, findet man es nicht. Jedenfalls nicht in der Form, die einem vorschwebt. Wie kriege ich das Plastik–Schächtelsen überhaupt auf? (Trick 17 mit dem Nipple durch die Lasche?) Wo finde ich Schaltpläne? Wo alles über den tatsächlich drinsteckenden Prozessor der Familie S1C33 (*Epson's original 32–bit RISC CPU*)? Wo stecken die SD–RAMs (bis zu 32 MB!)? Wo finde ich die zugehörigen Betriebs–Programme in einer Form, die mir zusagt? Wo finde ich *Kochrezepte* für den eventuellen Umbau? (So wie damals beim R8C mit pbForth und den Selbstbauextrasensoren.) Et cetera pp. Es geht nicht eigentlich um Wikipedia. Das brauche ich nicht so übermäßig oft. Und schon gar nicht in der Westentasche. Es geht um Selbstverwirklichung. Mit Mitteln, die man in der Bastlerkiste findet oder die man sich notfalls im Versandhandel für fast umsonst beschaffen kann. Es geht um Spaß an der Freude. Und was Forth betrifft, habe ich soviel in Erinnerung: An *Alles über eForth* (das steckt im Kasten drin) komme ich über den bekannten amerikanisch/chinesischen Forth–Enthusiasten C. H. Ting heran. Ich würde mich freuen, wenn sich der eine oder andere Forth–Freund für den WikiReader genau so begeistern könnte wie seinerzeit beim Lego–Roboter oder beim R8C/13. An meiner Mitwirkung soll es dann nicht fehlen. Was die Handhabung betrifft: Ganz so geschmeidig wie beim iPad läuft der fingerschiebebetriebene Touch–Screen nicht. Keine Hintergrundbeleuchtung, sehr kleine Buchstaben, nicht für jedes Auge geeignet. Wenn man auf ein Zeichen der eingeblendeten Tastatur (für Zweijährige oder Außerirdische) drückt, kommt das benachbarte Zeichen gleich mit. Na ja, aber *Openmoko* über *Openmoko* (Open Source) im Internet, mit der Erwartung, dass da für jeden Bastlergeschmack etwa dabei sein dürfte. Jedenfalls kommt schon mal Forth auf das Display, wenn man den Search–Knopf (Gummi) gleichzeitig mit dem Einschaltknopf (oben rechts) drückt. Im Manual (nur englisch) steht *WikiReader is meant to be used with adequate lighting.* Jedenfalls also keine vollmundigen Versprechungen. Man weiß, was einen erwartet: Ein Gerät eben zum Experimentieren.

Fred Behringer

Make:

Hallo.

Gerade eben habe ich nach dem in der letzten *Vierten Dimension* genannten US–Magazin *Make* geschaut und mir als allererstes diesen Film angesehen:

`http://blog.makezine.com/archive/2009/01/make_television_episode_3_steampunk.html`

Einfach phantastisch!

Es ist kein Wunder, dass die *DIY* (Do-It-Yourself) — Welle wieder voll im Kommen zu sein scheint. Die Leute haben weniger Geld — mit Ausnahme der Leute im Video, natuerlich.

Forth müsste eigentlich mit dabei sein, denn Forth ist die einzige DIY-Programmiersprache, die es gibt!

Mit den besten Grüßen aus Pensylvanien,

Dirk

amforth XXL

Matthias Trute

Amforth ist ein 16–Bit–Forth für die 8–Bit–Mikrocontrollerserie Atmega der Firma Atmel. Mit einer Cellsize von 16 Bit scheinen die Grenzen deutlich zu sein: 64 KB Adressraum. Doch da diese Controller einen getrennten Daten– und Programmadressraum haben (*Harvard Architektur*) steht der 64–KB–Adressraum zweimal zur Verfügung.

Tatsächlich läuft amforth jedoch auch auf Systemen, die deutlich mehr als dies haben. Und dieses Mehr steht den Programmen auch zur Verfügung, ohne dass man die Programme umschreiben müsste. Bei alledem belegt das Kernsystem nicht mehr Platz als auf den kleinen Systemen.

```
amforth 3.9 ATmega2561 14745 kHz
 > : hallo ." Hallo " ;
 ok
 > hallo
 Hallo  ok
 > words
 hallo d- d+ dinvert int@ int! is Rdefer
 Edefer show-wordlist ...
 > unused decimal u.
 57537  ok
 >
```

Um das zu erreichen, hat schon der Hersteller einige Entscheidungen beim Design der Controller getroffen, die amforth ausnutzt. Sie betreffen den Programmspeicher, der als selbst–reprogrammierbarer Flash ausgeführt ist und bei amforth das gesamte Dictionary enthält.

Die Speicheraufteilung

Amforth als 16–Bit–System ist natürlich begrenzt auf die Möglichkeiten, die 16 Bit bieten: 65536 Adressen. Beim Flash heisst das ungewohnterweise 128 KB. Der Grund ist folgender: Der Flash beinhaltet normalerweise den ausführbaren Code. Die Maschineninstruktionen der Atmegas sind nun immer 16 Bit oder Vielfache davon groß (RISC lässt grüßen). Atmel hat das so implementiert, dass hinter jeder Adresse 2 Bytes liegen. Dass Programme zwei Instruktionen benötigen (`lpm Z+`) und solcherart doch eine Art byte–basierte Adressierung zum Einsatz kommt, zählt wohl zu den Seltsamkeiten der Plattform...

Damit kann das Dictionary 128 KB groß werden. Bei Systemen, die noch mehr Flash haben, ergibt sich ein Dilemma: Der nutzbare Speicher ist kleiner als der vorhandene. Soll heißen: Es liegt Flashspeicher brach. Zum anderen ist der Sonderbereich, der für das Selbstprogrammieren des Flashes zu nutzen ist, jenseits der Adressgrenze für die zahlreich genutzten `JMP/CALL`–Befehle.

Letzteres ist einfach zu lösen: Es gibt einen neuen Befehl: `RAMPZ`. Er ist zwar deutlich umständlicher zu benutzen und damit langsamer, da aber der Flash–Programmierbefehl ohnehin 2–3 Millisekunden benötigt, spielt das keine Rolle.

Um ersteres Dilemma zu lösen, schließlich bezahlt niemand für etwas, was er nicht nutzen kann, gibt es mehrere Ansätze, die derzeit noch nicht umgesetzt sind, aber sicher über kurz oder lang nutzbar sein dürften: Zum einen kann man Worte definieren, die zur Adressierung

[1] `i!` ist das amforth–Wort, um in den Instruktionsspeicher zu schreiben. Hierbei ist die Adresse eine Zelle groß, also 16 Bit. `id@` liefert also keine doppelt–genaue Zahl, sondern nimmt eine solche als Adresse.

[2] Interessanterweise gibt es ein analoges Dilemma auch beim Zugriff auf SD–Karten: Das `BLOCKS` Wordset definiert Zugriffsworte, die genau eine Zelle als Blocknummer beinhalten. Dummerweise gibt es so kleine SD Cards aber gar nicht mehr... wenn man ein 16–Bit–Forth betrachtet.

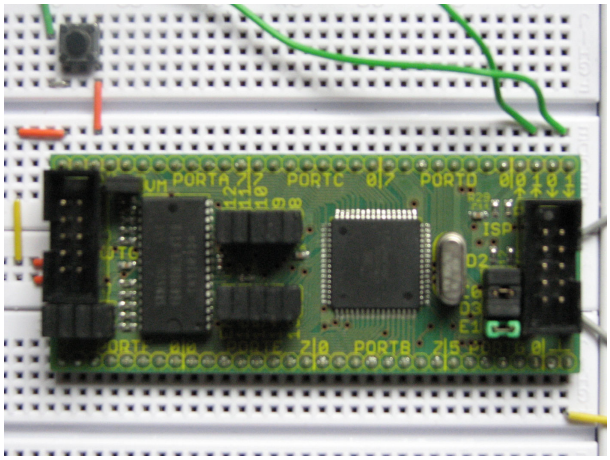


Bild 1: AVR–Entwicklungsmodul mit 128 KB ext. SRAM und ATMEGA2561 [3]

nicht eine, sondern zwei Zellen einsetzen. Dass das bislang nicht umgesetzt ist, ist auch eine Frage der Wortwahl: Wie sollte so ein Wort heißen? `id@ ( d -- n )` bzw. `id! ( n d -- )`?[1] [2]

Der zweite Ansatz geht auf eine Idee von Michael Kalus zurück: Flash als Blockdevice ansprechen. Und wieder gibt es zwei Varianten: Standard–Forth Blöcke von 1 KB oder die device–spezifische Pagesize nutzen. Relativer Nachteil der Standard–Blöcke ist, dass sie recht viel vom RAM für Buffer belegen, der immer knapp ist. Dies ist bei der zweiten Variante deutlich entspannter, dafür aber auch anspruchsvoller bei der Nutzung. Andererseits stehen beim Atmega2561 8 KB interner RAM zur Verfügung.

Technisches

Wer den Quellcode analysiert, wird feststellen, dass die Unterstützung für große (Flash–) Speichermengen in mehreren Stufen erfolgt. Die erste Stufe sind die 128–KB–Systeme. Hier kommt ein zusätzliches Adressregister RAMPZ zum Einsatz. Es erweitert das normalerweise genutzte Z–Registerpaar um ein weiteres Byte und stellt so die komplette Adresse bereit. Dieses Registertriplett wird von den Befehlen `elpm` und `spm` genutzt, um den Flashspeicher zu adressieren. Im Quelltext wird das alles in den Assemblermakros `readflashcell` und `writeflashcell` gekapselt. In diesen Macros ist auch die eigentümliche interne Zugriffsmethode auf Byteadressen versteckt.

Aus Performancesicht sind große Atmegas bei gleicher Taktfrequenz langsamer als kleinere. Kleine Systeme benötigen 4 Instruktionen, um eine Flashzelle zu lesen. Große Systeme müssen 7 Befehle ausführen, um das Gleiche zu leisten. Da jedoch zusätzlich immer die Stackoperationen und der innerere Interpreter involviert sind, ist die prozentuale Einbuße, bezogen auf das Gesamtsystem, deutlich geringer. In der Praxis wird man wohl einen etwas schnelleren Quarz verbauen...

```
.macro readflashcell
lsl zl
rol zh
lpm @0, Z+
lpm @1, Z+
.endmacro
.macro writeflashcell
lsl zl
rol zh
.endmacro
```

```
.macro readflashcell
clr temp7
lsl zl
rol zh
rol temp7
out RAMPZ, temp7
elpm @0, Z+
elpm @1, Z+
.endmacro
.macro writeflashcell
clr temp7
lsl zl
rol zh
rol temp7
out RAMPZ, temp7
.endmacro
```

Der scheinbar fehlende `spm`–Befehl im writeflashmacro ist an anderer Stelle enthalten.

Der zweite Schritt sind Systeme, die mehr als 128 KB haben. Hierzu zählt interessanterweise auch der ATXmega128. Diese Systeme sind erkennbar am 24 Bit (3 Byte) großen Program–Counter–Register. Damit legt z. B. ein CALL 3 Bytes auf den Returnstack, anstelle 2 bei den kleineren Controllern. Bei diesen Controllern liegt die Hürde darin, dass der Bereich, von dem aus das Flash–Selbstprogrammieren erfolgen muss, außerhalb der zulässigen Sprungdistanz der normalerweise genutzten Befehle liegt.

```
; ( n addr -- ) Memory
; R( -- )
; writes a cell in the flash
VE_DO_ISTORE:
    .dw $ff04
    .db "(i!)"
    .dw VE_HEAD
    .set VE_HEAD = VE_DO_ISTORE
XT_DO_ISTORE:
    .dw PFA_DO_ISTORE
PFA_DO_ISTORE:
  ....
  ldi zl, byte3(DO_ISTORE_atmega)
  out_ rampz, zl
  ldi zh, byte2(DO_ISTORE_atmega)
  ldi zl, byte1(DO_ISTORE_atmega)
  eicall ; reaches any address
  ....
  ; finally clear the stack
  loadtos
  rjmp DO_NEXT

.org NRWW_START_ADDR
  ; way outside of an call/jmp
DO_ISTORE_atmega:
  rcall pageload
  ; erase page if needed
  ; it is needed if a bit goes from 0 to 1
  com temp4
.....
  ret ; return to caller, all done
```

Damit amforth hier die nötige Flexibilität erhält, wurde das alles entscheidende Wort i! umgebaut, indem es in Assembler komplett neu geschrieben wurde. Das brachte neben einigen Bytes Platzersparnis auch die Möglichkeit, den Programmcode unabhängig vom Rest des Systems zu platzieren. Eine Alternative wäre, ein weiteres Forthsystem zu haben, das nur für die Ausführung von `i!` zuständig ist.

Damit rückt auch ein weiteres Ziel in Reichweite: Da es nur wenige Bytes umfasst, kann man eine Einbeziehung in Standard–Bootloader angehen, die dann nicht nur ein neues System einspielen können, sondern darüberhinaus noch eine API für Programme wie amforth anbieten können, um den Flash gezielt ändern zu können.

Der Inhalt des Bootloaderbereichs ist für das amforth–System immer read–only. Daraus ergibt sich die Konsequenz, dass amforth diesen Bereich maximal nutzen sollte. Bei den Systemen mit bis zu 128 KB Flash heisst dies: Alles dort ablegen, was irgendwie geht. Die Systeme mit mehr Flashspeicher sind da exakt entgegengesetzt:

Da amforth mit den Standardworten den Bootloaderbereich nicht addressieren kann, darf *nichts dort liegen. Allerdings* muss das Wort `i!` dort platziert werden, da für die Dauer des Flashprogrammierens unter

keinen Umständen auf den RWW–Flash, der das Dictionary enthält, zugegriffen werden darf. Die CPU wird in diesem Fall einfach gestoppt. Um das zu lösen, liegt der Dictionary–Eintrag von `i!` zwar im normalen Flash, ruft aber Code im Bootloaderbereich auf. Dieser Code kehrt erst dann zurück, wenn die Arbeit getan ist. Wie man an dem obigen Codeschnipsel sehen kann, ist die Implementierung nicht für jedes Wort sinnvoll einzusetzen.

Zukünftiges

Prognosen sind schwierig, vor allem, wenn sie die Zukunft betreffen. Die Eckdaten für amforth werden wohl zukünftig so aussehen: Das Dictionary umfasst maximal 128 KB.

Jeglicher Flashspeicher jenseits der 128 KB wird z.B. als Block über eine 16 Bit große Blocknummer angesprochen. Alternativ oder zusätzlich kann ein Wortpaar analog zu `i@` und `/i!` existieren, das als Adresse eine doppelt–genaue Zahl hat und so den Zugriff auf den gesamten Flashspeicher gewährt.

Links

[1] `http://www.atmel.com/products/AVR/`
[2] `http://amforth.sourceforge.net/`
[3] Anbieter des Entwicklungsboard: `www.alvidi.de`

Gehaltvolles

zusammengestellt und übertragen von *Fred Behringer*

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 79, April 2010

Das gesamte diesmalige Vijgeblaadje besteht aus dem folgenden ausführlichen Artikel:

Ombouw printerdriver kastje tot een general purpose computer

Robert Henneke

Der Autor schreibt:

„Worüber geht es?

Vor etwa einem Jahr erfreute uns Ron Minke mit einer Anzahl von Printer-Driver-Kästchen mit der Bemerkung: Tut damit, was ihr wollt. Ich beschloss, daraus einen Allzweck-Computer zu bauen, vergleichbar mit einem PC."

Und weiter:

„Was bezwecken wir damit?

Das Printer-Driver-Kästchen enthält eine Platine, auf welcher ein klassischer 8052-Computer sitzt: Ein 80C32-Prozessor, 32-KB-EPROM, 32-KB-RAM, RS-232-Interface, Centronics-Printer-Port — und noch etwas Peripherie. Der 80C32 hat kein internes ROM, ist also vom Programm im EPROM abhängig. Im vorliegenden Fall ein angepasstes ATS-Forth. Jedes andere Programm würde also die Anfertigung eines neuen EPROMs bedeuten. Wir möchten einfach nur ein kleines Programm ins RAM platzieren. Ersetzen wir aber das EPROM durch RAM, dann geht nichts mehr, da ja dann nach dem Einschalten kein *Boot-Programm* mehr zur Verfügung steht."

Der 80C32 wird durch einen AT89S8252 ersetzt. Dieser ist mit dem 80C32 Pin- und Funktions-kompatibel und hat außerdem 8 KB an (Flash-)CODEROM und 2 KB EEPROM. Ins CODEROM wird ein *Operating-System* gelegt, das mit dem BIOS eines PCs verglichen werden kann.

Der Umbau

Der AT89S8252 wird über das Monitor-Programm FLIPMON52 programmiert. Es arbeitet anfangs mit dem EPROM. Weiter: Den Inhalt des EPROMs (das ATS-Forth) sichern - Die Latches - Die Verbindung mit der Außenwelt - ... Der Rest des Berichts wurde vom Autor auf der Allgemeinen Mitgliederversammlung am 10. April 2010 vorgetragen.

Das Vijgeblaadje und die HCC-Forth-gg

Das Vijgeblaadje erscheint um den Ersten eines jeden geraden Monats herum. Neues von der Forth-gebruikersgroep erfährt man per `http://www.forth.hcc.nl/nieuws`.

Bootmanager und FAT–Reparatur: Siebter Fort(h)schritt (PC–Bootdisk)

Fred Behringer

Ziel

Es wird ein *minimalistischer* Bootmanager entwickelt, der von Diskette aus die gewünschte Partition der Boot–Festplatte aus deren Partitionstabelle interaktiv auszuwählen gestattet. Die im Listing zu findende *Organisation* umfasst etwa 10 Zeilen Forth. Außerdem wird auf der Diskette (natürlich) ein Forth–System (hier Turbo–Forth) mit dem einbezogenen Wort bootpart aus Teil 1 und dem neuen Forth Wort pc–hd verlangt. Die Anpassung an ZF und andere Systeme bleibe zur Aufgabe gestellt. Auch beim Übergang von Diskette zu Stick oder CD sehe ich keine unüberwindlichen Schwierigkeiten — soweit der PC das Booten von den genannten Medien überhaupt erlaubt.

Wie?

Viele Bootmanager nisten sich auf einem *garantiert* nicht verwendeten Platz der Boot–Festplatte ein. Bei dem von mir seit Jahren verwendeten XFDISK von Florian Painke und Ulrich Müller sind es die ersten 17 Sektoren nach dem MBR. Warum Bootdiskette? Das von den diversen Windows–Versionen her bekannte Prinzip der Bootdiskette (eigentlich nur für Reparaturen vorgesehen) erzwingt sich zunächst einmal das Booten des PCs unter DOS (im BIOS als erstes Bootmedium einzustellen). Auf der Bootdiskette kann ich den gesamten (auf DOS basierten) Forth–Überbau zur Organisation des Bootmanagers in Anspruch nehmen. Das gibt mir viele leicht zugängliche Möglichkeiten für eigene Konstruktionen. Alles, was zur Auswahl der gewünschten Boot–Partition x nötig ist, ist eigentlich nur die Anpassung der Aktiv–Kennzeichnung (80 statt 00) im Offset 1be+((x-1)*10) und der Markierung zum Verstecken (beispielsweise 06 statt 16) im Offset 1c2+((x-1)*10) der Partitionstabelle des MBRs (LBA–Sektor 0), mit entsprechenden Rücksetzungen in der Herkunfts–Partition. Anschließend stellt man den PC im BIOS auf Booten von Festplatte um und schaltet ihn kurz aus und dann wieder ein.

Schwierigkeiten

macht der Umstand, dass es heutzutage zwar üblich ist, im BIOS die Sequenz *Erstes Bootmedium Diskette, zweites Bootmedium Festplatte* einstellen zu können, dass sich aber der PC beim anschließenden Rebooten mit herausgenommener Diskette aufhängt. Das scheint besonders dann der Fall zu sein, wenn es sich um ein USB–Diskettenlaufwerk handelt. Ansonsten könnte man den PC mit immer derselben BIOS–Einstellung zur Boot–Reihenfolge (erstens Diskette, zweitens Festplatte) laufen lassen und bräuchte nur darauf zu achten, die Bootdiskette nach der Auswahl eines anderen Betriebssystems (einer anderen Partition) und vor dem Rebooten des PCs aus dem Laufwerk zu nehmen.

Zur Abhilfe

habe ich vor einiger Zeit irgendwo einen Tipp darüber gelesen, mit welch einfachem Patch des Bootsektors der betreffenden Diskette man das BIOS zwingen kann, bei nicht eingelegter Diskette einfach klagmeldungslos zum zweiten Bootmedium (der Festplatte) überzugehen, und aber an der im BIOS eingestellten Bootreihenfolge nichts zu ändern. Ich kann mich nicht mehr erinnern, wo ich das gelesen habe, und schlage vor, mit dem Googlen beispielsweise unter dem Suchbegriff Partition–Image Boot–Floppy–Maker 1.0 anzufangen.

Zur Erläuterung

des im vorliegenden Artikel Gesagten hier die Partitionstabelle des MBRs (ab Offset 1be), die bei mir folgende Gestalt hat:

```
80 01 01 00 06 FE 3F 3F 3F 00 00 00 01 B0 0F 00
00 00 01 40 05 FE FF FF 40 B0 0F 00 33 BD 04 0B
00 00 C1 FF 17 FE FF FF 73 6D 14 0B 27 B8 1A 06
00 00 C1 FF 1C FE FF FF 9A 25 2F 11 27 B8 1A 06
```

Die primäre Partition 1 und die auf der Partition 2 (erweiterte Partition) eingerichteten logischen Laufwerke D: bis S: laufen durchgängig als FAT16 und sind DOS 6.2, Windows 3.11, Windows 95 und Windows 98 vorbehalten (momentan an dem hier verwendeten Experimentiersystem nur DOS verwirklicht). Auch Linux (UBUNTU – 64 Bit) soll auf die erweiterte Partition (als *logisches Laufwerk* unter extfs2) gelegt werden. Die primäre Partition 3 ist mit XP–Home belegt (NTFS). Die primäre Partition 4 beherbergt Windows–ME (FAT32).

Viel zu umständlich?

Ja schon, wenn man an die komfortablen professionellen Bootmanager (Beispiel GRUB) denkt. Andererseits hätte ich mich (als Amateur) nie ohne Forth an das Thema Bootmanager gewagt. Und welche Möglichkeiten ich da habe, an dem hier gezeigten Minimalmanager nach Belieben und eigenem Gusto herumzuexperimentieren! Und unter welch enormem Durchblick! Ganz anders als bei den *Professionellen*. Und wenn der *normalerweise* eingesetzte Bootmanager vertrauenswürdiger Herkunft (bei mir bisher XFDISK) versagt, ist es beruhigend zu wissen, dass die hier gezeigten einfachen Forth–Mittel vielleicht auch schon weiterhelfen können — bevor man die diversen *Reparatur–Programme* durchprobiert, die erklärungslos losrattern und das Durcheinander eventuell nur noch vergrößern.

Die Stabilität der Boot–Reihenfolge

mag bei dem hier gezeigten Minimalsystem noch ausbaubedürftig sein. Auf jeden Fall ist aber sichergestellt, dass eine einmal vorgenommene Umstellung der Festplatten–Partitionstabelle per `boot pc-hd` alle System–Ausschaltungen übersteht. Es kann also bei einem Mehrbootsystem schon mal nicht mehr passieren, dass man an sein System plötzlich überhaupt nicht mehr herankommt — ob Windows oder nicht Windows.

Natürlich auch mit XP

Was wird in den Computer–Blättern nicht alles über *XP und Linux gleichzetig* oder *XP und ME gleichzeitig* oder *DOS(FAT16) und XP(NTFS) gleichzeitig* gesagt! Es erübrigt sich zu bemerken, dass die Einbindung von Windows–XP in den vorliegenden rudimentären Selbstbau–Bootmanager eine der leichtesten Übungen der Welt ist.

Vorgehensweise

Das von mir zum Experimentieren und Ausprobieren zugrundegelegte Szenario geht aus dem Listing mit den \–Kommentaren hervor. Das Bootmenü (zur wahlweisen Aktivierung der einen oder anderen Partition in der Partitionstabelle des MBRs) habe ich `pc-hd` genannt, damit das (interaktive) *Booten* von der Diskette aus mit der sich einprägenden Sequenz `boot pc-hd` erledigt werden kann. Für das Betriebssystem der Bootdiskette habe ich FreeDOS gewählt, um zusätzlich zum DOS 6.2 auf der Festplatte noch ein weiteres DOS–System zur Auswahl zu haben (Auswahl 0 = Beim Disketten–System bleiben). In `boot` versteckt sich das Turbo–Forth–System, das um das Programm aus Teil 1 der vorliegenden Artikelserie und das hier besprochene Boot–Menü erweitert und dann per `save-system` abgespeichert wurde. Sein voller DOS–Programm–Name lautet `boot.com` und das Forth–Wort `pc-hd` wird bei Aufruf `boot pc-hd` als DOS–Parameter–String übergeben. Genau da liegt der wunde Punkt beim Versuch, das Ganze auf ZF zu übertragen. Ich gebe diesen Versuch für den Augenblick auf und habe vor, mich später wieder damit zu beschäftigen. Irgendwie muss bei jedem neuen Bootvorhaben dafür gesorgt werden, dass im BIOS die Bootreihenfolge *Disk,HD* eingestellt ist (oder bleibt). Die in der Partitionstabelle jeweils eingestellte Bootpartition bleibt nach dem Ausschalten des PCs bestehen. Will man danach wieder mit dieser schon eingeschalteten Bootpartition arbeiten, dann sorge man einfach dafür, dass der PC nicht von Diskette, sondern von HD bootet. Andernfalls wieder wie oben.

Überraschung mit den Laufwerkbuchstaben

Beim Experimentieren mit meinem System (siehe Listing) wurde bei Auswahl 3 (Windows–XP) während der zwischenzeitlichen Boot–Einstellung das DOS–Laufwerk D: nach C: verschoben. Und alle weiteren logischen Laufwerke entsprechend. Das hat seine Richtigkeit: XP läuft unter NTFS und wird von der DOS–Bootdisk einfach ignoriert. Anders bei Auswahl 4 (Windows–ME). Da wurde C: von ME (zu meiner Überraschung) auch für das DOS–Interim der Bootdisk zum (durchaus nicht ignorierten) C: . Auch das hat seine Richtigkeit: FreeDOS kann mit FAT32 umgehen — und mein ME läuft unter FAT32.

Listing: PC-HD.FTH

```
1   \ ************************************************************************
2   \ *                                                                      *
3   \ * pc-hd.fth                                                            *
4   \ *                                                                      *
5   \ * Disketten-Bootmanager zum Booten einer PC-HD unter Turbo-FORTH-83 *
6   \ *                                                                      *
7   \ * Fred Behringer     -     Forth-Gesellschaft      -     01.09.2010 *
8   \ *                                                                      *
9   \ * Aufruf: boot pc-hd                                                   *
10  \ *                                                                      *
11  \ ************************************************************************
12
13  \ PC-HD soll heissen:
14  \ Zugrunde gelegt wird ein 80x86-Rechner mit der Bootfestplatte HD.
15
16  \ 'boot' ist das anzufertigende DOS-Programm boot.com (siehe gleich).
17
18  \ 'pc-hd' ist das neu praesentierte, unten stehende Boot-Menue.
19
20  \ Anfertigung von boot.com (fuer Turbo-Forth-16-Bit):
21  \ Forth aufrufen.
22  \ include xxx , wobei xxx.fth das Forth-Programm-Paket aus Teil 1 ist.
23  \ include pc-hd , wobei pc-hd.fth das vorliegende Programm ist.
24  \ save-system boot.com.
25
```

```
26  \ ZF legt der Anfertigung von boot.com ein paar Huerden in den Weg. Ich
27  \ ueberlasse es den ZF-Fans unter den Lesern zur Uebung, eine Anpassung
28  \ vorzunehmen.
29
30  hex
31
32  : pc-hd ( -- )
33     getmbr
34     begin
35       begin
36         cr ." [0]: FreeDOS von Boot-Diskette"
37         cr ." [1]: DOS 6.2 von Festplatte"
38         cr ." [3]: Windows-XP"
39         cr ." [4]: Windows-ME" cr
40         cr ." [*]= Die Taste * druecken!" key
41         dup 30 = if drop 0 true else
42         dup 31 = if drop 1 true else
43         dup 33 = if drop 3 true else
44         dup 34 = if drop 4 true else drop false then then then then
45       until
46       cr cr ." Gebootet wird [" dup . bs emit ." ]"
47          cr ." OK? (j/n)" key ascii j = if true else drop false then
48     until
49     dup 0 = if bye then bootpart
50     ." Diskette (USB-Laufwerk?) entfernen und PC aus/einschalten/! ..."
51     key ;
```

Fingerübungen in Forth: parsing

Anton Ertl, Michael Kalus

Neulich dachten wir darüber nach, wie die Datei `m168def.inc` mit Forth interpretiert werden könnte. Diese Datei wird vom Assembler AVRA benutzt, um allen Registern und besonderen Bits des atmega168 einen Namen zu geben. Für einen Crosscompiler müssten diese Konstanten übernommen werden. Hier ist ein Beispiel aus der Datei:

```
1   #ifndef _M168DEF_INC_
2   #define _M168DEF_INC_
3   #pragma partinc 0
4   ; ***** SPECIFY DEVICE
5   ************************************************
6   .device ATmega168
7   #pragma AVRPART ADMIN PART_NAME ATmega168
8   .equ    SIGNATURE_000   = 0x1e
9   .equ    SIGNATURE_001   = 0x94
10  .equ    SIGNATURE_002   = 0x06
11  #pragma AVRPART CORE CORE_VERSION V2E
12  ; ***** I/O REGISTER DEFINITIONS
13  ****************************************
14  ; NOTE:
15  ; Definitions marked "MEMORY MAPPED"are extended I/O ports
16  ; and cannot be used with IN/OUT instructions
17  .equ    UDR0    = 0xc6  ; MEMORY MAPPED
18  .equ    UBRR0H  = 0xc5  ; MEMORY MAPPED
19  .equ    UBRR0L  = 0xc4  ; MEMORY MAPPED
20  .equ    UCSR0C  = 0xc2  ; MEMORY MAPPED
21  .equ    UCSR0B  = 0xc1  ; MEMORY MAPPED
22  .equ    UCSR0A  = 0xc0  ; MEMORY MAPPED
23  .equ    TWAMR   = 0xbd  ; MEMORY MAPPED
24  .equ    TWCR    = 0xbc  ; MEMORY MAPPED
25  .equ    TWDR    = 0xbb  ; MEMORY MAPPED
```

```
26   .equ    TWAR    = 0xba  ; MEMORY MAPPED
27   ..
```

Um Fehler zu vermeiden, soll die Datei nicht verändert werden. Wie können also aus den `.equ`–Konstanten nun Forth–Konstanten werden?

Erster Versuch von Michael

So eine `.equ`–Zeile hat Ähnlichkeit mit einem Forth–Ausdruck:

```
28     : QWER 0x1122 ; \ ich bin ein Kommentar
```

Drum versuchte er, die Zeilen zu interpretieren, indem er diese Ähnlichkeit abbildete.

```
29   vocabulary converter   converter definitions
30   : _;        [compile] ;                 ; immediate
31   : ;         [compile] ; [compile] \    _; immediate
32   : .equ      [compile] :                _; immediate
33   : =                                    _; immediate
34
35   .equ    QWER = 0x1122 ; this is a comment.
```

Führt man nun `QWER` aus, findet man `0x1122` auf dem Stack und der Kommentar ist weg. Das geht also. Aber warum geht das? Und gibt es einen anderen Weg?

Antons Lösungen

Zunächst zeigte er, dass der Ansatz, die Ähnlichkeit zu verwenden, auch noch anders gemacht werden kann.

```
1     wordlist constant converter-wordlist
2     get-current converter-wordlist set-current
3     : ; postpone ; 0 parse 2drop ; immediate
4     : .equ : ;
5     : = ; immediate
6     set-current
7     converter-wordlist >order
8     .equ    QWER = 0x1122 ; this is a comment.
9     previous
```

Doch ist man damit nicht flexibel genug, auch andere Konstrukte aus der Datei zu übersetzen. Zeilenweise zu parsen, scheint da besser zu sein. Und man wird die Kommentare eleganter los.

```
10    : .equ : parse-name 2drop ( the "=" ) parse-name evaluate postpone ; ;
11    : ; source nip >in ! ;
12    .equ    QWER = 0x1122 ; this is a comment.
```

Auch diese Fassung sollte, so wie die vorherige, in eine eigene Wordlist gesetzt werden, weil man sonst keine Colon–Definitionen mehr beenden kann. Probiert's mal aus. Viel Spaß beim Knobeln.

Quelle: comp.lang.forth

Der Wikireader

Carsten Strotmann

Es kommt selten vor, dass ein elektronisches Gerät für sogenannte “Endbenutzer” (also Oma, Opa, Mama, Papa, Du und ich) als OpenSource veröffentlicht wird. Nicht nur die Software (Firmware), sondern auch die komplette Hardware ist offengelegt und dokumentiert. Noch ungewöhnlicher ist es, dass ein solches Gerät ein Forth enthält. Der WikiReader ist ein solches Gerät.

Wikipedia in der Tasche

Vordergründig ist der WikiReader ein Gerät, mit dem man die Inhalte der Wikipedia (`http://wikipedia.org`) “offline” mitführen kann. Das funktioniert auch sehr gut, so eignet sich der Wikireader als Nachschlagehilfe auf Studienreisen oder im Zug. Ideal auch für Themenanregungen, wenn dem Besitzer des WikiReaders auf Partys beim Smalltalk die Themen ausgehen (der “random”-Knopf ist hier hilfreich).

Der Wikireader (9.5cm × 9.5cm × 2cm, 140g) liegt gut in der Hand. Angeschaltet wird das Gerät über einen Knopf oben rechts, unterhalb des Touchscreens befinden sich drei Gummiknöpfe, “search”, “history” und “random”. Nach dem Anschalten erscheint nach wenigen Sekunden eine Tastatur auf dem Schwarz-weiß-Bildschirm, über die der Benutzer einen Suchbegriff eingeben kann. Der Bildschirm (240 ×208 Bildpunkte) ist ein kapazitiver Touch-Screen, welcher über ein asynchrones serielles Protokoll an die Wikireader-Hauptplatine angebunden ist. Dieser Bildschirm ist nicht entspiegelt und nicht beleuchtet, hat aber einen guten Kontrast und lässt sich daher auch bei Sonnenschein (am Strand) benutzen. Ein Druck auf die “Search”-Taste durchsucht die Wikipedia Artikel (ca. 6 GB Daten bei der englischen Ausgabe) in Sekundenschnelle und zeigt eine Liste der gefundenen Wikipedia-Artikel. Diese können über den Touchscreen ausgewählt werden.

Ein Artikel kann mit einer vertikalen Wischbewegung über den Bildschirm gescrollt werden. Hyper-Links werden wie im Webbrowser unterstrichen dargestellt. Leider zeigt der Wikireader derzeit nur den Text der Artikel an, jedoch keine Grafiken. Tabellen werden angezeigt, sind aber bei großen Tabellen schwer zu lesen. Die Wikireader-Software wird weiterentwickelt und zukünftige Versionen sollen auch Bilddaten zeigen können.

Das Innenleben

Im Inneren des Wikireaders werkelt mit dem Epson S1C33E07 ein relativ unbekannter Single-Chip-Mikrokontroller. Der Epson ist Mitglied der S1C33-RISC-Familie, besitzt 8kB und 2kB internen RAM-Speicher, einen LCDKontroller mit eigenem Frame-Buffer, serielle Schnittstellen, SPI- sowie GPIO- und eine Keypad-Schnittstelle. Damit befindet sich schon fast alles, was der WikiReader benötigt, in dem Mikrokontroller. Zusätzlich befindet sich auf der Platine im Inneren des WikiReaders noch zusätzlich 32MB SDRAM (16bit Bus, 4 Bänke) und das Boot-ROM (SPI, 64KB Flash).

Abbildung 1: Wikipedia in der Westentasche: der Wikireader

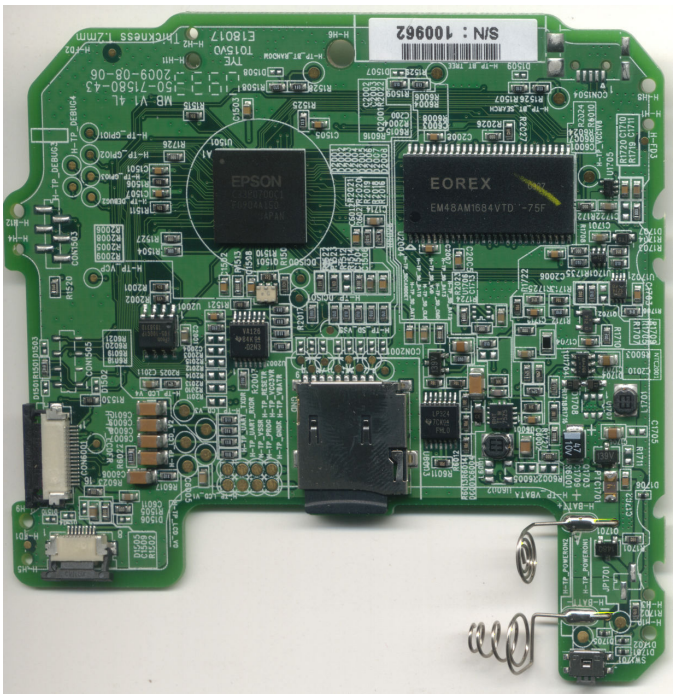


Abbildung 2: Die Wikireader–Systemplatine

Die Wikipedia Daten befinden sich auf einer Micro-SD- (oder Micro-SDHC bis 8GB, VFAT formatiert)-Karte innerhalb des Batteriefachs (2 x AAA, mein Wikireader läuft nun über 6 Monate mit der ersten Batterieladung). Diese Karte kann vom Benutzer entnommen werden und über einen PC oder Mac können die Wikipedia-Daten und auch die Wikireader-Firmware aktualisiert werden. Offizielle Updates gibt es halbjährlich zum kostenlosen Download von der WikiReader-Webseite. Da es sich um ein OpenSource-Produkt handelt, finden sich im Internet an verschiedenen Stellen alternative Firmware und Wikipedia-Daten (z. B. Pakete mit den deutschen Wikipedia-Inhalten). In der Zukunft wird der Wikireader auch E-Books anzeigen können, eine Vorabversion mit Büchern aus dem Projekt Gutenberg (`http://www.gutenberg.org`) ist seit Juli 2010 verfügbar.

Wikireader und OpenMoko

Der Wikireader wird von der taiwanesischen Firma OpenMoko hergestellt und ist ein aus der Not geborenes Produkt. OpenMoko ist in der FreeSoftware-Gemeinschaft insbesondere für das FreeRunner-Mobiltelefon bekannt, der Versuch, ein Mobiltelefon ausschließlich aus OpenSource-Bestandteilen zu erstellen. Beim FreeRunner-Projekt gab es allerdings unvorhergesehene Probleme, und es wurde kein für Endanwender benutzbares Produkt innerhalb des geplanten

Abbildung 3: Das OpenMoko-Handy und der Wikireader

Zeitrahmens fertiggestellt. Das Geld wurde knapp, und OpenMoko musste ein Produkt auf den Markt bringen, um weitere OpenSource-Entwicklungen finanzieren zu können. Dieses Produkt ist der WikiReader.

Wikireader und Forth

Die Hauptanwendung des WikiReaders, das Anzeigen der Wiki-Artikel, ist in C geschrieben. Forth benutzen die Entwickler des WikiReaders für die Fehlersuche. Glücklicherweise wird das Forth im WikiReader nicht versteckt, sondern ist dokumentiert und kann vom Benutzer aufgerufen werden.

Wird der WikiReader bei gedrückter "search"-Taste angeschaltet, so startet anstatt der normalen WikiReader-Software ein Menüprogramm, welches die Forth-Programme auf der SD-Karte auflistet. Der Benutzer kann nun per Touchscreen diese Programme auswählen und starten. Mitgeliefert werden Beispielprogramme für die Ansteuerung des Bildschirms, Abfrage des Touch-Screens und der Tasten sowie Zugriff auf den Massenspeicher (SD-Karte).

Das Forth im WikiReader ist ein in C implementiertes und ANSIfiziertes eForth, portiert vom OpenMoko-Entwickler Christopher Hall. Der Quellcode des Forth-Systems befindet sich im Unterverzeichnis "/samo-lib/forth" auf GitHub. Die Beispielprogramme finden sich ebenfalls im GitHub unter "/samo-lib/forth/programs".

Hier ein Beispielprogramm, ein simples Malprogram für den Touchscreen (line.4th):

```
\ ctp line drawing
base @ decimal
variable down
: line-draw ( -- )
    lcd-cls
    s" Line Drawing" lcd-type
    lcd-cr lcd-cr
    s" Touch screen to draw line" lcd-type
    9 lcd-text-rows 1- lcd-at-xy
    s" Clear" lcd-type
    10 lcd-spaces
    s" Exit" lcd-type
    button-flush
    key-flush
    ctp-flush
    false down !
    lcd-black
    begin
        ctp-pos? if
            ctp-pos dup 0<
            if
                2drop
                false down !
            else
                down @
                if
                    lcd-line-to
                else
                    lcd-move-to
                    true down !
                then
            then
        then
        button? if
            button
            case
                button-left of
                    lcd-cls
                    false down !
                endof
                button-centre of
                endof
                button-right of
                    exit
                endof
            endcase
        then
        key? if
            key-flush
        then
        wait-for-event
    again
;
base !
```

Im Batteriefach des Wikireaders befinden sich Lötpunkte für die serielle Schnittstelle. Über diese Schnittstelle kann das Forth im WikiReader interaktiv benutzt und programmiert werden. Alternativ können die Programme auf einem PC erstellt und dann auf die SD-Karte übertragen werden.

Fazit

Der WikiReader ist zum Preis von 99 US$ (ca. 76 Euro) ein interessantes Spielzeug für den Forth-begeisterten Hobby-Tüftler. Obwohl ich bei der Wikipedia-Funktion des WikiReaders erst skeptisch war ("braucht man das?") und ich den WikiReader wegen des Forth-Systems gekauft habe, kann ich sagen, dass sich der WikiReader gerade auf Reisen als nützlich erwiesen hat. Das Open-Source-Konzept und das Forth-System im WikiReader laden zum Basteln ein.

Links

[1] Wikireader - `http://thewikireader.com`
[2] Wikireader Entwickler - `http://dev.thewikireader.com`
[3] Quellcode auf GitHub - `http://github.com/wikireader`

Protokoll der Mitgliederversammlung 2010

Carsten Strotmann

Rostock, 28.3.2010 10:00 Start der Mitgliederversammlung der Forth–Gesellschaft e.V. 2010

Es sind 25 Mitglieder von 119 Mitgliedern anwesend
Als Versammlungsleiter wird Martin Bitter gewählt Die Beschlussfähigkeit wird festgestellt

Bericht vom Webmaster (Ulrich Hoffmann)

- Web-Präsenz ist stabil, Webserver–Software wird ggf. aktualisiert
- es werden mehr Artikel für die Hauptseite benötigt

Zeitschrift Vierte Dimension

- es werden mehr Artikel benötigt
- Fred Behringer und Michael Kalus helfen (internationale Kontakte, Leserbriefe, Zusammenfassung comp.lang.forth)
- Applaus für Ulli und das Redaktionsteam

Ergänzungen zur Tagung

- Diskussion über Tagung 2011: Preis, Ort, Datum

Bericht des Direktoriums

- Mitgliederentwicklung: 8 Austritte, 5 Neuaufnahmen - 119 Mitglieder im Dezember 2009
- Finanzamt hat Gemeinnützigkeit geprüft, die Steuerbefreiung wurde am 18.11.2009 erteilt, Steuerbefreiung für max. 5 Jahre erhalten, keine Beanstandungen
- Angaben zur zeitnahen Mittelverwendung des Vereinsvermögens (ca. 18.300,00 Euro) musste bis 15.12.2009 nachgereicht werden
- Auflösungsteil § 16 der Satzung vom 18.04.2004 entspricht nicht mehr geltendem Recht seit 01.01.2007. Nach neuster Rechtslage ist es nicht mehr zulässig, die konkreten Angaben über die anfallberechtigte juristische Person bzw. steuerbegünstigte Körperschaft offenzulassen.

Kassenbericht (Ewald Rieger)

- Einnahmen aus ideellem Bereich
  - Mitgliedsbeiträge: 2.290,00
  - Mitgliedsbeiträge Folgejahr: 16,00
  - Spenden: 8,00
  - Summe: 2.314,00
- Ausgaben aus ideellem Bereich
  - Mitgliederpflege
    - Kosten der Mitgliederverwaltung: -815,66
    - Büromaterial: -78,32
    - Porto, Telefon: -10,55
    - Einzugskosten: -8,50
  - Summe Kosten der Mitgliederpflege: -913,03
    - Forth–Projekte–Unterstützung: -644,30
    - Reisekostenerstattungen: -717,90
    - Repräsentationskosten: -242,16
  - Summe Veranstaltungen: -960,06
    - Geschenke, Jubiläen, Ehrungen: -38,25
    - Internetkosten: -452,11
  - Summe Kosten ideeller Bereich: -3.007,75
- Vermögensverwaltung:
  - Einnahmen der Vermögensverwaltung
    - Zinserträge 0 % USt: 182,99
  - Kosten der Vermögensverwaltung
    - Nebenkosten des Geldverkehrs: -197,26
  - Summe Vermögensverwaltung: -14,27
- Zweckbetrieb:
  - Einnahmen aus Zweckbetrieben
    - Einnahmen Zeitschrift VD 7%: 1.847,28
    - Umsatzsteuer 7 %: 129,72
    - USt-Erstattung Vorjahr: 273,51
  - Summe Einnahmen aus Zweckbetrieben: 2.250,51
  - Kosten des Zweckbetrieb
    - Zeitschrift Vierte Dimension Druckkosten: -1188,95
    - Transportkosten VD: -641,85
    - Verpackungskosten VD: -29,37
    - Verwaltungskosten: -462,76
    - Abziehbare Vorsteuer 19%: -319,41
  - Summe Kosten aus Zweckbetrieben: -2642,34
- Einnahmen / Ausgaben:
  - Einnahmen ideeller Bereich: 2.314,00
  - Kosten ideeller Bereich: -3.007,75
  - Vermögensverwaltung: -14,27
  - Einnahmen aus Zweckbetrieb: 2.250,51
  - Kosten aus Zweckbetrieb: -2.642,34
- Verlust zum 31.12.2009: -1.099,85

Wirtschaftsplan 2010 (Ewald Rieger)

- Einnahmen
  - Mitgliedsbeiträge: 2200
  - Zweckbetrieb VD: 1900
  - MWST Erstattung: 189
  - Summe: 4289
- Ausgaben
  - Mitgliederpflege: 900
  - Internetkosten: 450
  - Vermögensverwaltung: 200

  - VD 4/2009: 650
  - VD 1-4/2010: 2500
  - Unterstützung Projekte: 700
  - Forth-Tagung 2010: 2000
  - Linuxtag: 1500
  - Lizenzen für Bücher: 1500
  - Summe: 10400
- Plan 2010:
  - Summe Einnahmen: 4289
  - Summe Ausgaben: -10400
  - Verlust: -6111
- Diskussion/Abstimmung über Wirtschaftsplan wurde unter Verschiedenes verschoben

Bericht des Kassenprüfer (Peter Kalkert)

- es wurden keine Unstimmigkeiten festgestellt

Bericht über Projekte (Bernd Paysan)

- Linuxtag 2010
- Beagleboard (ARM Forth auf TI Beagleboard)
- Bücherprojekt

Entlastung des Direktoriums:

- 0 Enthaltungen
- 0 Gegenstimmen
- 25 Stimmen für Entlastung

Wiederwahl Direktorium

Kandidaten: Ulrich Hoffmann, Bernd Paysan, Ewald Rieger

- 0 Enthaltungen
- 0 Gegenstimmen
- 25 Stimmen für Wiederwahl

Satzungsänderung §16

- Bei Auflösung der Forth–Gesellschaft soll das Restvermögen einem gemeinnützigen Verein zufallen
- Es wurden Vorschläge für gemeinnützige Vereine gesammelt, Vorstellung der Vorschläge und Abstimmung:
  - Ärzte ohne Grenzen: 3
  - AATis e.V.: 13
  - Brot für die Welt: 0
  - FSFE (Free Software Foundation Europe): 8
  - Greenpeace: 0
  - Toppoint e.V.: 0
- Text in der Satzung §16 wird geändert:
  - alter Text: “Bei Auflösung des Vereins oder bei Wegfall steuerbegünstigter Zwecke fällt das Vermögen des Vereins an eine juristische Person des öffentlichen Rechts oder eine andere steuerbegünstigte Körperschaft zwecks Verwendung für Wissenschaft und Forschung.”
  - ersetzt durch: “Bei Auflösung des Vereins oder bei Wegfall steuerbegünstigter Zwecke fällt das Vermögen des Vereins an *Arbeitskreis Amateurfunk in der Schule AATis e.V.*, der es ausschließlich und unmittelbar für gemeinnützige Zwecke zur Förderung von Wissenschaft und Forschung oder Bildung zu verwenden hat.”
  - Abstimmungsergebnis Satzungsänderung:
    * 0 Enthaltungen
    * 25 Ja-Stimmen
    * 0 Gegenstimmen

Projekte

- Schulungsprojekt gibt Übersicht der Aktivitäten in der VD
- Abstimmung finanzielle Unterstützung des Schulungsprojektes:
  - 2 Enthaltungen
  - 23 Ja-Stimmen
  - 0 Gegenstimmen

Verschiedenes

- Tagung
  - Diskussion über Preise
  - Richtschnur ca. 250 Euro (Einzelzimmer + Tagungsbeitrag)
  - Vorschlag für 2011: Goslar Zeppelinhaus
  - Organisation: Thomas Prinz
  - Termin: März (Ostern) 2011
- Abstimmung über den “Wirtschaftsplan 2010” der Verwaltung (Ewald Rieger)
  - 3 Enthaltungen
  - 0 Gegenstimmen
  - 22 Ja-Stimmen

12:15 Ende der Mitgliederversammlung

Halbduplex–Multidrop–Stromschleife und amforth

Michael Kalus

Für eine Solarkollektoranlage [1] zur Warmwasserbereitung wurden neulich drei Steuerungseinheiten mit je einem atmega168 eingesetzt. In der Anlage waren 12 Temperaturen zu erfassen, 3 Pumpen und 8 Ventile zu schalten. Da preiswerte fertige Steuerungseinheiten zur Verfügung standen, die jeweils 4 Temperaturwerte erfasssen und über 4 Triacs 220V–Verbraucher schalten konnten, wurden drei davon in der Anlage verteilt. Damit konnte zum einen die Steuerungsaufgabe gelöst, und außerdem die verlegten Kabelmeter gering gehalten werden.

Die drei *Dosen* genannten Steuerungen und ein Terminal [2] wurden über eine V24–20mA–Stromschleife miteinander verbunden und halbduplex betrieben. Die Speisung der Stromschleife erfolgte über eine kleine separate Konstantstromquelle. Die *Dosen* und das Terminal wurden über Optokoppler [3] mit der Stromschleife verbunden. Die Verbindung vom USART des atmega in jeder *Dose* hin zu den Optokopplern erfolgte also vollduplex. Erst die Optokoppler schalten das auf eine Leitung, wodurch die Halbduplex–Eigenschaft der Verbindung entsteht. An den Steuerungen mussten somit keine Hardwareänderungen vorgenommen werden. Es genügte eine Anpassung des amforth an den Halbduplex–Effekt.

Vom Terminal aus gesehen: Fernes Forth Echo nicht erwünscht

Gewöhnlich wird amforth in einer Vollduplex–Verbindung mit einem Terminal eingesetzt, also Punkt–zu–Punkt. Es gibt einen Hinkanal vom Terminal zum amforth, das die Zeichen mit dem Wort `KEY` empfängt. Und einen Rückkanal, über den amforth per `EMIT` Zeichen an das Terminal sendet. amforth wertet alle mit `KEY` empfangenen Zeichen im Wort `ACCEPT` aus, von dem aus dann der Terminal Input Buffer `TIB` gefüllt und ein Echo jedes Zeichens an das Terminal zurückgegeben wird (remote echo). Da nun das Terminal selbst halbduplex an die Stromschleife angeschlossen ist, erscheint sofort auch das von ihm selbst gesendete Zeichen auf dem Schirm — sein eigenes lokales Echo. So kommt es, dass bei jedem Tastendruck das Zeichen gleich zweimal auf dem Bildschirm erscheint. Und da das lokale Echo seitens des Terminalanschlusses bei dieser Schaltung nicht unterdrückt werden kann, muss das Echo im amforth entfernt werden. Die Stellen, an denen das `ACCEPT` dafür verändert werden muss, sind im Listing wiedergegeben.

Vom amforth aus gesehen: Eigenes lokales Echo wegwerfen

Damit das, was amforth selbst sendet, nicht sogleich wieder in seinem eigenen `TIB` landet und interpretiert wird — um also eine Rückkopplung der eigenen Ausgabe zu verhindern — muss das unvermeidliche amforth–seitige lokale Echo vernichtet werden (echo cancellation). Unvermeidlich, weil man den USART der Einfachheit halber einfach weiter arbeiten lässt. So empfängt eine *Dose* zwar sein soeben gesendetes Zeichens selbst, dieses wird aber sofort wieder verworfen. Das ist der einfachste Weg, das lokale Echo unschädlich zu machen. Außerdem kann man dabei auch schnell noch nachsehen, ob die Leitung frei war, das Zeichen also ungestört emittiert werden konnte.

```
: drop-echo-emit ( c -- )
   tx begin key? until key drop ; ↩

' drop-echo-emit is emit ↩
```

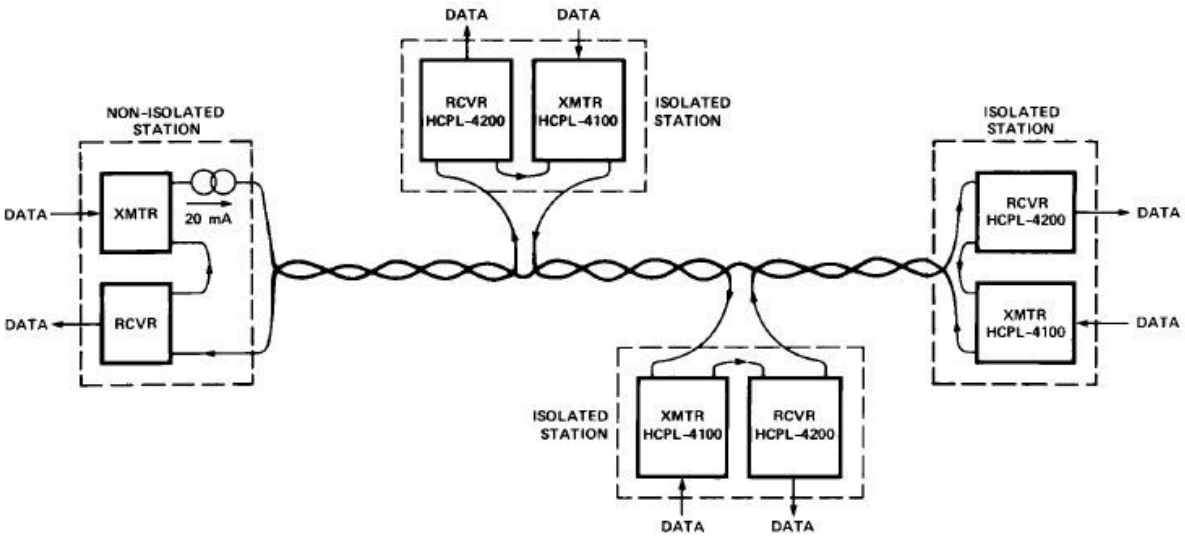


Abbildung 1: Schema der Stromschleife und ihrer Optokoppler

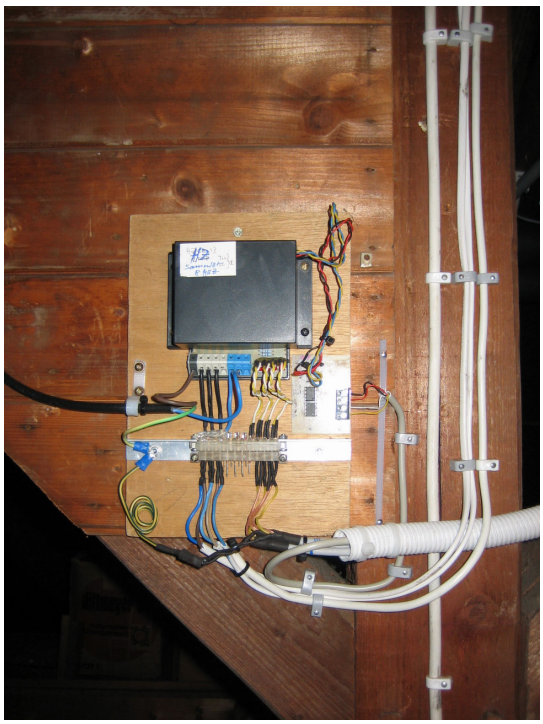


Abbildung 2: Eine *Dose* auf dem Dachboden

Prompt gestalten

Da schon das Terminal mit seinem abschließenden CR und auto–LF den Cursor selbst in die nächste Zeile setzt, ist das zusätzliche CR des amforth, das mit seinem OK kommt, störend. Außerdem sollte jede *Dose* ihr eigenes Zeilenprompt haben. So wurde unterschieden zwischen einem OK–Prompt, das gesendet wird, wenn der Interpreter mit der Verarbeitung seines letzten `TIB` fertig ist, und einem Zeilen–Prompt, das erscheint, wenn der Eingabestrom abgearbeitet wurde. Dieses Verhalten ist im Wort `QUIT` angesiedelt. Dem `QUIT` wurden nun zwei deferred words an die Stelle der eingebauten Ausgabe mitgegeben, `OK_PROMPT` und `READY_PROMPT`. Damit kann der Prompt umgestellt werden, je nach Zweck und *Dose*. Mit diesen Anpassungen ließen sich die Steuerungen an der Stromschleife nun ordentlich einzeln ansprechen und halbduplex betreiben. Das Listing des `QUIT` zeigt die Stellen, an denen die Änderungen für Verhalten des Prompt eingefügt wurden. Außerdem ist gezeigt, wie vom Assemblerlisting aus ein deferred word `PROMPT` angelegt wird.

Das Protokoll, mit dem nun alle *Dosen* im Ring kommunizieren können, wird in einem weiteren Beitrag beschrieben.

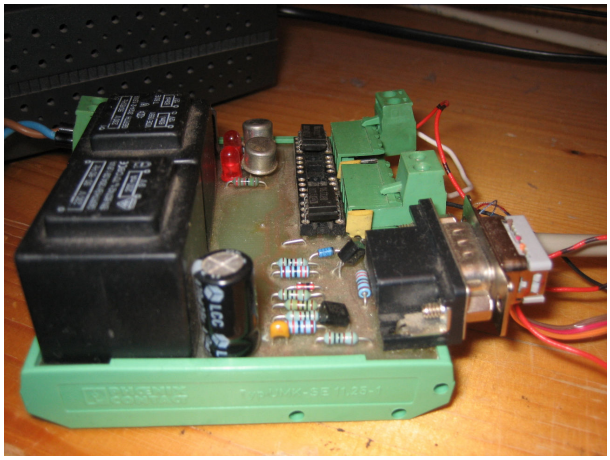


Abbildung 3: Konstantstromquelle für die Stromschleife mit seriellem Anschluss des Terminals

Quellen

[1] Vacuumtubes als Solarkollektoren, 3 Einheiten zu je 22 Röhren, die einen Sammler heizen. http://www.wattsolar.de/

[2] Das Terminal besteht lediglich aus einem USB–seriellen Wandler, auf den ein Terminalprogramm auf einem Laptop wirkt (ZTerm auf MacOSX).

[3] Optically Coupled 20 mA Current Loop Transmitter HCPL–4100, and Optically Coupled 20 mA Current Loop Receiver HCPL–4200 von Hewlett Packard. In dem Dokument ist auch die Konstantstromquelle für die Stromschleife angegeben.

Assemblerlisting des amforth–Files `accept.asm` mit den Modifikationen

```
1   ; ( addr n1 -- n2 ) System
2   ; R( -- )
3   ; reads a line with KEY into addr until n1 characters are reveived or cr/lf detected.
4   VE_ACCEPT:
5       .dw $ff06
6       .db "accept"
7       .dw VE_HEAD
8       .set VE_HEAD = VE_ACCEPT
9   XT_ACCEPT:
10      .dw DO_COLON
11  PFA_ACCEPT:
12      .dw XT_DUP          ; ( -- addr n1 n1)
13      .dw XT_TO_R
14      .dw XT_TO_R
15  PFA_ACCEPT1:            ; ( -- addr )
16      .dw XT_KEY          ; ( -- addr k )
17      .dw XT_DUP          ; ( -- addr k k )
18      .dw XT_DOLITERAL
19      .dw 10
20      .dw XT_NOTEQUAL
21      .dw XT_DOCONDBRANCH         ; case linefeed -> done&exit
22      .dw PFA_ACCEPT2
23      .dw XT_DUP
24      .dw XT_DOLITERAL
25      .dw 13
26      .dw XT_NOTEQUAL
27      .dw XT_DOCONDBRANCH         ; case cr -> done&exit
28      .dw PFA_ACCEPT2
29      ; check backspace
30      .dw XT_DUP
31      .dw XT_DOLITERAL
32      .dw 8
33      .dw XT_EQUAL
34      .dw XT_DOCONDBRANCH         ; case BS do the backspace
35      .dw PFA_ACCEPT3
36      ; delete previous character
37      ; check beginning of line
38      .dw XT_R_FROM             ; ( -- addr k n1 )
39      .dw XT_R_FETCH            ; ( -- addr k n1 n2)
40      .dw XT_OVER               ; ( -- addr k n1 n2 n1)
41      .dw XT_TO_R
42      .dw XT_EQUAL              ; ( -- addr k f )
43      .dw XT_DOCONDBRANCH         ;
44      .dw PFA_ACCEPT5
45      ; we are at the beginning of the line, ignore this character
46      .dw XT_DROP               ; ( -- addr )
47      .dw XT_DOBRANCH             ; recurse
48      .dw PFA_ACCEPT1
49  PFA_ACCEPT5:                  ; backspace handdling in Terminal
50  .if (halfduplex==1)
```

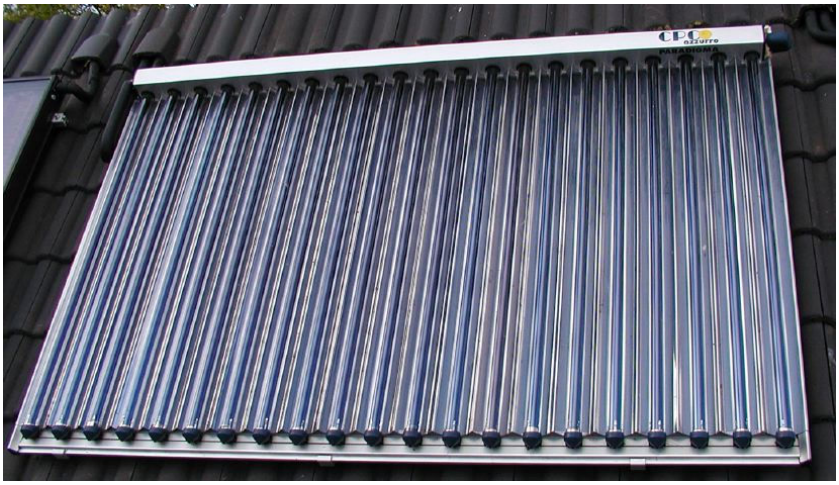
```
51      ; emit nothing = echo cancellation; BS is localy echod in the ring.
52  .else
53      ; emit the key = echo
54      .dw XT_DUP                  ; ( -- addr k k )
55      .dw XT_EMIT                 ; ( -- addr k ) ; BS
56  .endif
57      .dw XT_SPACE                ; ( -- addr k ) ; space
58      .dw XT_EMIT                 ; ( -- addr )   ; BS
59      .dw XT_1MINUS               ; ( -- addr--)
60      .dw XT_R_FROM
61      .dw XT_1PLUS
62      .dw XT_DOBRANCH               ; goto
63      .dw PFA_ACCEPT4
64  PFA_ACCEPT3:
65      ; check for remaining control characters, replace them with blank
66      .dw XT_DUP              ; ( -- addr k k )
67      .dw XT_BL
68      .dw XT_LESS
69      .dw XT_DOCONDBRANCH
70      .dw PFA_ACCEPT6
71      .dw XT_DROP
72      .dw XT_BL
73  PFA_ACCEPT6:                      ; send echo:
74  .if (halfduplex==1)
75      ; emit nothing = echo cancellation.
76                              ; ( -- addr k)
77  .else
78      ; emit the key = echo
79      .dw XT_DUP              ; ( -- addr k k)
80      .dw XT_EMIT             ; ( -- addr k)
81  .endif
82                                  ; now store the key to buffer
83      .dw XT_OVER             ; ( -- addr k addr
84      .dw XT_CSTORE           ; ( -- addr)
85      .dw XT_1PLUS            ; ( -- addr++)
86      .dw XT_R_FROM           ; ( -- addr n1)
87      .dw XT_1MINUS           ; ( -- addr n1--)
88  PFA_ACCEPT4:
89      .dw XT_DUP
90      .dw XT_TO_R
91      .dw XT_EQUALZERO
92      .dw XT_DOCONDBRANCH         ; If recurse
93      .dw PFA_ACCEPT1
94      .dw XT_DUP
95  PFA_ACCEPT2:                      ; get number of tokens in TIB
96      .dw XT_SLASHKEY
97      .dw XT_DROP
98      .dw XT_DROP
99      .dw XT_R_FROM
100     .dw XT_R_FROM
101     .dw XT_SWAP
102     .dw XT_MINUS
103 .if (halfduplex==1)
104     ; noop
105 .else
106     .dw XT_CR
107 .endif
108     .dw XT_EXIT
```

Assemblerlisting des amforth–Files quit.asm mit den Modifikationen

```
1   ; ( -- ) System
2   ; R( -- )
3   ; main loop of amforth. accept - interpret in an endless loop
4   VE_QUIT:
5       .dw $ff04
6       .db "quit"
7       .dw VE_HEAD
8       .set VE_HEAD = VE_QUIT
9   XT_QUIT:
10      .dw DO_COLON
11  PFA_QUIT:               ; init main front loop:
12      .dw XT_SP0          ;   init stacks
13      .dw XT_SP_STORE
14      .dw XT_RP0
15      .dw XT_RP_STORE
16      .dw XT_LBRACKET     ;   zero state !
17  PFA_QUIT2:              ; check state:
18      .dw XT_STATE
19      .dw XT_FETCH
20      .dw XT_EQUALZERO    ; 0= if prompt else refill then
21      .dw XT_DOCONDBRANCH ;
22      .dw PFA_QUIT4
23  .if (halfduplex==1) ; final prompt on new line.
24      .dw XT_READY_PROMPT
25  .else
26      .dw XT_CR
27      .dw XT_SLITERAL
28      .dw 2
29      .db "> "
30      .dw XT_ITYPE
31  .endif
32  PFA_QUIT4:                      ; refill
33          .dw XT_STARTTERMINAL    ; deferred word: noop default. set to xon.
34      .dw XT_REFILL
35          .dw XT_STOPTERMINAL     ; deferred word: noop default. set to xoff.
36      .dw XT_DOCONDBRANCH         ; refillflag=true  if recurse
37      .dw PFA_QUIT2
38      .dw XT_DOLITERAL            ; else interpret then
39      .dw XT_INTERPRET
40      .dw XT_CATCH
41      .dw XT_QDUP
42      .dw XT_DOCONDBRANCH         ; if ok recurse
43      .dw PFA_QUIT3
44      .dw XT_DUP                  ; exception
45      .dw XT_DOLITERAL
46      .dw -2
47      .dw XT_LESS
48      .dw XT_DOCONDBRANCH         ; if error" ??" tib position
49      .dw PFA_QUIT5
50  .if (halfduplex==1)
51          ; noop
52  .else
53          .dw XT_CR    ; mk
54  .endif
55          .dw XT_SLITERAL
56          .dw 4
57          .db  " ?? "
58          .dw XT_ITYPE
59          .dw XT_BASE
```

```
60          .dw XT_FETCH
61          .dw XT_TO_R
62          .dw XT_DECIMAL
63          .dw XT_DOT
64          .dw XT_G_IN
65          .dw XT_FETCH
66          .dw XT_DOT
67          .dw XT_R_FROM
68          .dw XT_BASE
69          .dw XT_STORE            ; then
70          .dw XT_CR    ; mk
71  PFA_QUIT5:
72          .dw XT_DOBRANCH     ; restart
73      .dw PFA_QUIT
74  PFA_QUIT3:                  ; ok recurse
75  .if (halfduplex==1)         ; call deffered prompt
76      .dw XT_OK_PROMPT
77  .else
78  ;    .dw XT_CR
79      .dw XT_SLITERAL
80      .dw 5
81      .db "  ok ",0
82      .dw XT_ITYPE
83  .endif
84      .dw XT_DOBRANCH
85      .dw PFA_QUIT2
86      .dw XT_EXIT ; never reached
87
88
89
90  .if (halfduplex==1)  ; deferred prompt.
91
92  ; ( -- n*y ) System Value
93  ; R( -- )
94  ; deferred action
95  VE_OK_PROMPT:
96      .dw $ff09
97      .db "ok_prompt"
98      .dw VE_HEAD
99      .set VE_HEAD = VE_OK_PROMPT
100 XT_OK_PROMPT:
101     .dw PFA_DODEFER
102 PFA_OK_PROMPT:
103     .dw EE_OK_PROMPT
104     .dw XT_EDEFERFETCH
105     .dw XT_EDEFERSTORE
106
107 ; ( -- n*y ) System Value
108 ; R( -- )
109 ; deferred action
110 VE_READY_PROMPT:
111     .dw $ff0C
112     .db "ready_prompt"
113     .dw VE_HEAD
114     .set VE_HEAD = VE_READY_PROMPT
115 XT_READY_PROMPT:
116     .dw PFA_DODEFER
117 PFA_READY_PROMPT:
118     .dw EE_READY_PROMPT
119     .dw XT_EDEFERFETCH
```

```
120         .dw XT_EDEFERSTORE
121
122     ; default ok prompt:
123     ; ( -- )
124     VE_OPROMPT:
125         .dw $ff07
126         .db "oprompt",0
127         .dw VE_HEAD
128         .set VE_HEAD = VE_OPROMPT
129     XT_OROMPT:
130         .dw DO_COLON
131     PFA_OPROMPT:
132         .dw XT_CR
133         .dw XT_SLITERAL
134         .dw 4
135         .db " ok "
136         .dw XT_ITYPE
137         .dw XT_EXIT
138
139     ; default ready prompt:
140     ; ( -- )
141     VE_RPROMPT:
142         .dw $ff07
143         .db "rprompt",0
144         .dw VE_HEAD
145         .set VE_HEAD = VE_RPROMPT
146     XT_RPROMPT:
147         .dw DO_COLON
148     PFA_RPROMPT:
149         .dw XT_CR
150         .dw XT_SLITERAL
151         .dw 2
152         .db "» "
153         .dw XT_ITYPE
154         .dw XT_EXIT
155     .endif
156
157     ; use:
158     ; ' oprompt is ok_prompt
159     ; ' rprompt is ready_prompt
160
161     ; finis
```



Ein Vakuumröhrenkollektor, Quelle: wikimedia.org

Forth im Garten

Rolf Schöne

In einem Garten gräbt man um, düngt, sät, pflanzt, erntet, und das war's. Das war's? Nein, ab und zu (wöchentlich!) muss man auch den Rasen mähen. In einem Garten lebt man naturnah. Naturnah? Nein, viel Energie muss man zum Tee- und Kaffekochen in Form von Gasflaschen importieren.

In einem Garten lebt man ohne Strom. Ohne Strom? Das geht. Aber wenn man einen akkubetriebenen Rasenmäher hat, eine solche Bohrmaschine dazu, und auch noch einen solchen Trimmer, dann möchte man den Transport von Akkus nachhause (zum Aufladen) schon gerne vermeiden. Es gibt doch eine ausgereifte Solar-Technik. Und die installierte ich [1].

Und wenn man schon Solarstrom hat, dann hänge ich doch auch noch eine Kühlbox dran. Aber Vorsicht: Die zieht satte 5 Ampère aus einem Akku mit 85 Ah. Für 17 Stunden würde das theoretisch reichen - wenn der Akku voll geladen ist. Ist er das nicht und hat man keinen Tiefentladeschutz (wie ich), dann ist der Akku schnell kaputt.

Es schadet ihm auch, wenn man nicht berücksichtigt, dass er als Solarakku mit seiner Plattengeometrie für einen Dauerstrom von 0,85 A ausgelegt ist (und nicht für 5 A). Schade, der war mit 300 € ja nicht gerade billig. Was tun?

Googeln. Und da findet man eine Anleitung zum Regenerieren von Akkus [2]. Also, versuche ich das und baue mir nach diesem Rezept einen Desulfator [3]. Nur musste ich täglich 22 km radeln, um ihn morgens einzuschalten und abends wieder auszuschalten, damit eine neue Tiefentladung des Akkus während der Nacht nicht geschehen kann.

Nun hat Erich Wälde in der VD 3,4/2006 und 1/2007 gezeigt, dass man mit dem Forth von Heinz Schnitter und Bernd Paysan (VD 2/2006 S.16f.) auf dem R8C/13 so einiges machen kann. Zwei selbstgestrickte R8C/13-Bretter liegen noch rum, also gehe ich dran, meine Forth-Kenntnisse nach langer Zeit aufzufrischen (das ging ziemlich flott) und das Ein- und Ausschalten des Desulfators zu automatisieren.

Warnung: Dies ist ein typisches Anfänger-Projekt und einer ambitionierten Zeitschrift wie dieser eigentlich nicht würdig. Profis mögen weiterblättern. Aber vielleicht gibt es doch Interessenten, die einen ersten Einstieg in Forth anhand eines ganz einfachen Problems suchen.

Bild 1 zeigt die Schaltung, die parallel zum Programm 1 entwickelt wurde. Das ist in zwei halben Tagen erledigt. Das Programm könnte durch ein paar Konstanten mehr noch lesbarer formuliert werden.

Die *Auftragslage* ist klar: Ein Desulfator ist abhängig von Tageszeit und Ladezustand des Akkus in einen bestehenden Ladekreis einzubinden. Möglichst lange soll er pro Tag aktiv sein.

Eine *Problemanalyse* war schon lange im Kopf, nur war ich mir nicht gewiss, ob eine hochgenaue Referenzspannung benötigt wird. Ich baue sie auf, verwende sie aber nicht sofort.

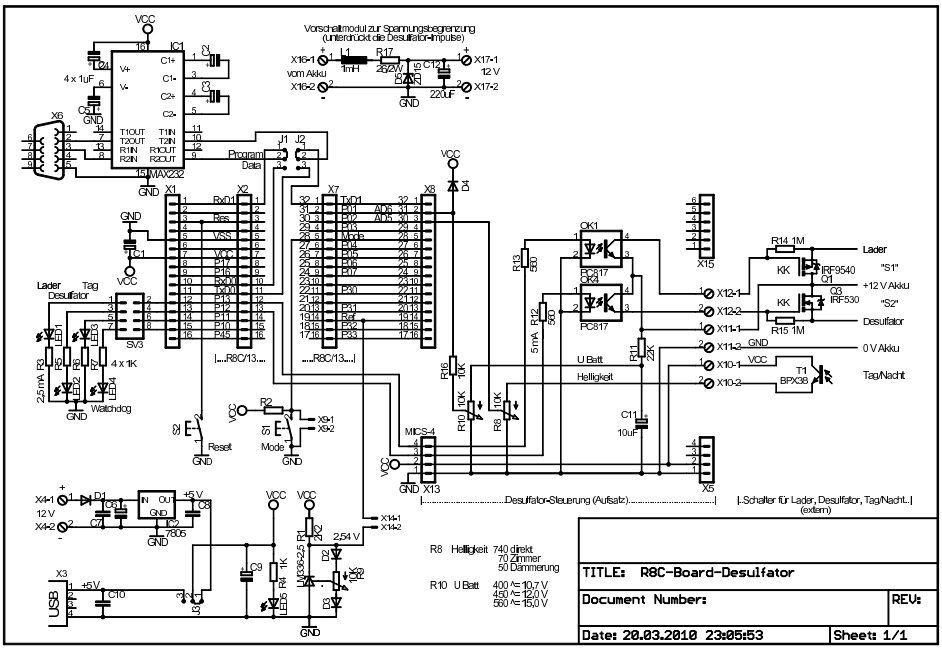


Bild 1: Schaltung der Desulfator–Steuerung

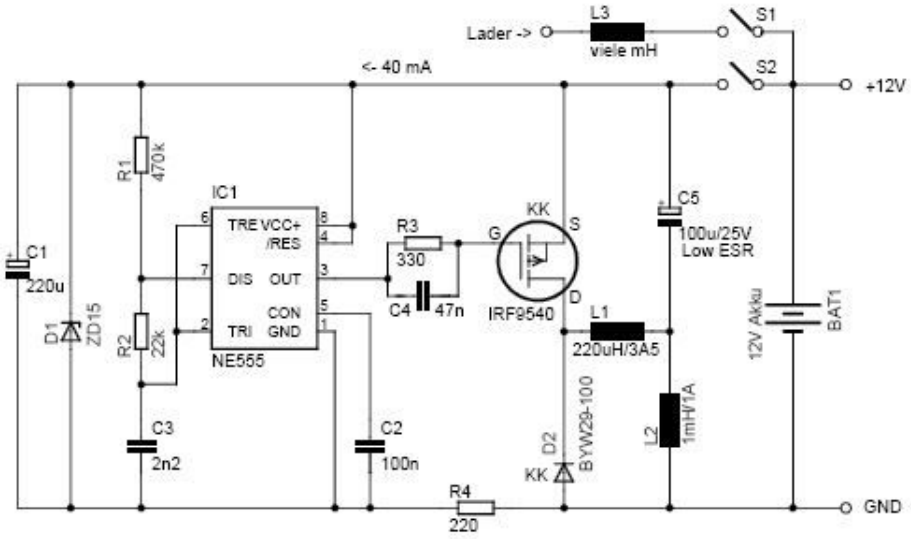


Bild 2: Schaltung des Desulfators

Die *Detailentscheidungen* sind anhand des Datenblattes sehr schnell zu treffen. Beispiel: Eine Zustandsanzeige muss sein, die Ports P10..13 bieten sich dafür an und sind auch schon vorverdrahtet. Statt separate Ports zur Steuerung der Optokoppler zu verwenden und deren Zustand auf P10..13 zu spiegeln, verwende ich diese Ports für beide Zwecke und schlage zwei Fliegen mit einer Klappe. Können die Ports doch laut Datenblatt 40 mA treiben - so viel ist hier gar nicht nötig, 2 mA reichen aus.

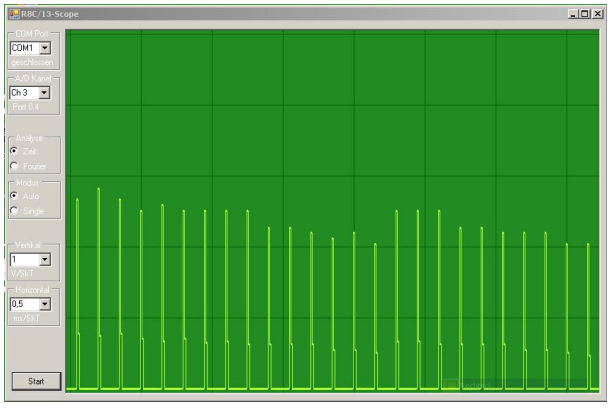
Die Löterei und die Installation brauchten weitere drei halbe Tage, und der Test braucht keine Zeit, lief die Steuerung doch auf Anhieb. Mit einem benachbarten Mittelwellen-Radio konnte ich das erwartete Kilohertz hören (ein alter Trick aus den Tagen, als Rechner noch begehbar waren).

Jetzt wollte ich die Ladeimpulse des Desulfators aber auch auf einem Oszilloskop sehen. Da kommt mir das zweite brachliegende R8C/13-Brettl gerade recht.

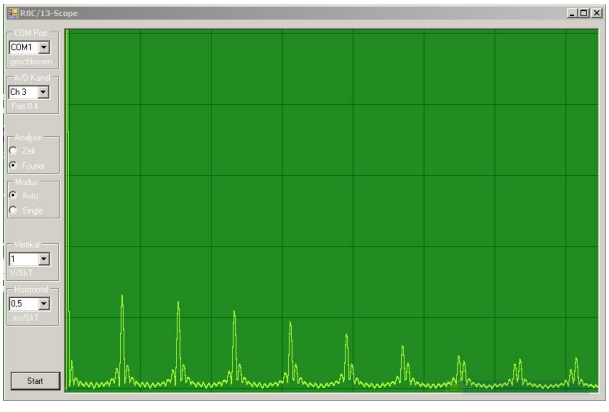
Wenn man nur schätzen kann, welche Pegel zu erwarten sind, dann klemmt man sie zum Schutz der Eingänge über einen Vorwiderstand vorsichtshalber an VCC, und nachdem Induktivitäten im Spiel sind, dann klemmt man sie besser auch noch GND. Bild 3 auf der nächsten Seite zeigt die Schaltung. Auf die Buchsen X8 und X9 kann man eine Eichquelle setzen, deren Schaltung hier nicht gezeigt wird. Das Programm 2 liefert Daten, in diesem Fall an VisualBasic, welches das Ergebnis auf einem Laptop anzeigt.

VisualBasic mag ich nicht, und Win32Forth [4] bereitet mir noch Schwierigkeiten bei der grafischen Darstellung.

Hier das Ergebnis der Messung:



Impulse von 3,5 A erwartete ich und war enttäuscht. Aber denkt man sich den Messwiderstand von 2,2 Ohm weg, dann kommt das schon hin.



Die Fourier-Analyse. Dieser Ripple scheint darauf hin zu deuten, dass Verbände von Bleisulfat-Kristallen zertrümmert werden (zu beweisen!).

Und so kam Forth-Schritt endlich auch in unseren Garten, und das gleich zweifach.

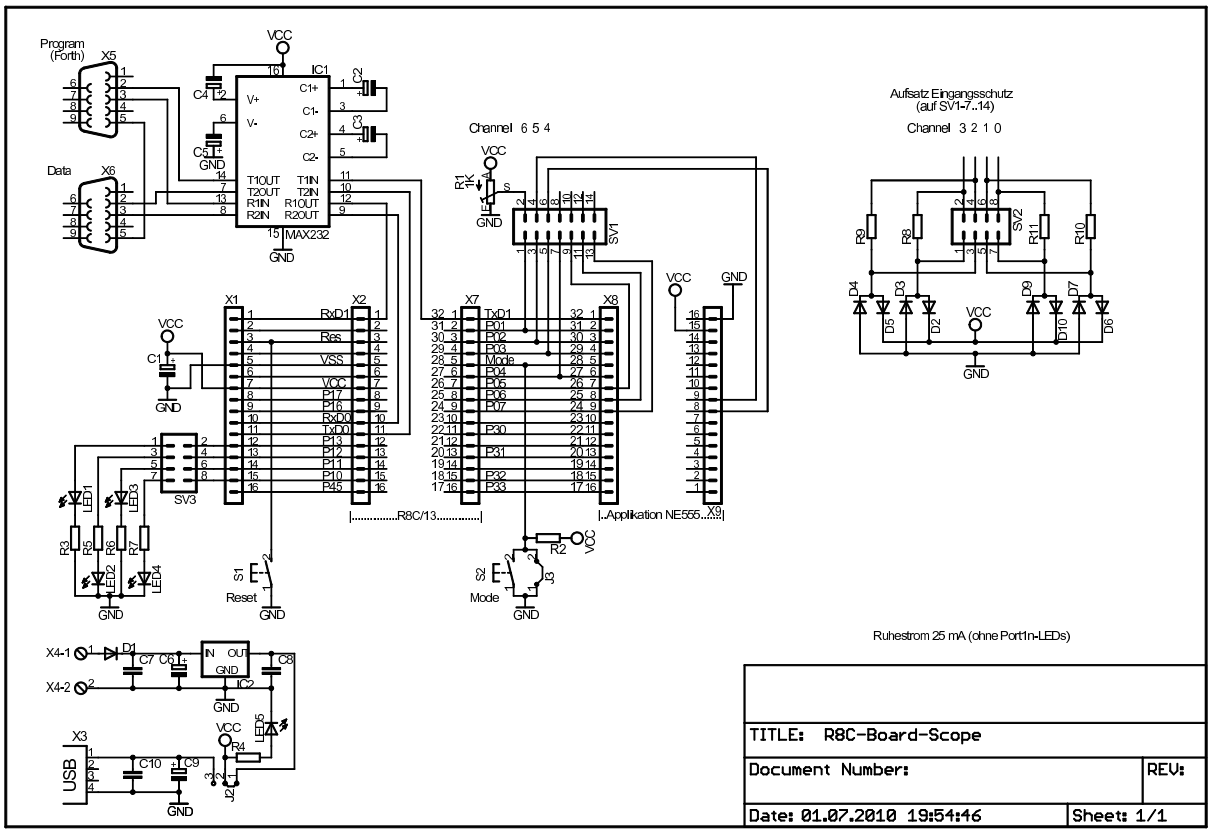
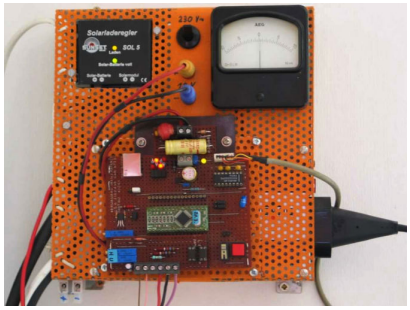


Bild 3: Schaltung des Oszilloskops

Quellen

[1] http://www.rolf-schoene.de/html/fotostrom.html
[2] http://www.members.shaw.ca/pferlow/desulfator.htm
[3] http://www.rolf-schoene.de/html/desulfator.html
[4] http://win32forth.de.vu/



Programm 1 — Desulfator.fs

```
1  \ Desulfator.fs       Stand 2010-03-20
2  \ Desulfator-Steuerung mit R8C/13   rs
3
4  ram
5
6  : Doit ;                \ for MARKER
7
8  %00000010 constant PD0   \ PRC2-bit
9  $0A       constant PREG  \ Protect Register
10 %00000001 constant TxD1  \ bit 0 is Output, bits 1..7 are ADC-Inputs
11 $E2       constant PDIR  \ Direction Register port0
12 $E5       constant port3 \ P30..P33 Input (not yet used)
13
14 variable Akku
15
16 rom
17
18 : AdcInit ( -- )         \ override lcdinit
19    PD0 PREG bset         \ enable PD0 in Protect Register
20    TxD1 PDIR c!          \ bit 0 is TxD1 Output,
21    &20 ms                \ bit 1..7 is Input Channel 6..0, settle
22 ;
23
24 : Tag? ( -- f )                       \ ist Tag?
25    1 port1 btst                       \ test bit 1
26 ;
27
28 : Tag! ( -- )                         \ ist Tag!
29    1 port1 bset                       \ set bit 1
30 ;
31
32 : Nacht! ( -- )                      \ ist Nacht!
33    1 port1 bclr                       \ clear bit 1
34 ;
35
36 : Wdog! ( -- )                       \ init Watchdog to off
37    0 port1 bclr                       \ clear bit 0
38 ;
39
40 : Wdog ( -- )                        \ Watchdog
41    port1 c@ $0F and 1 xor led! &10 ms  \ toggle bit 0 and wait 10 ms
42    Wdog!                              \ clear bit 0 again
43 ;
44
45 : 10s ( -- )                         \ 10 Sekunden warten und blinken
46    Wdog!
47    &10 0 do Wdog Wdog &980 ms loop
48 ;
49
50 : Hell ( -- )                        \ Messe Helligkeit, Channel 5
51    5 adc@ &50 >
52    if Tag! else Nacht! then           \ >50 (Daemmerung) ist Tag
53 ;
54
55 : Des-an ( -- )                      \ Desulfator an
56    2 port1 bset                       \ set bit 2
57 ;
58
59 : Des-aus ( -- )                     \ Desulfator aus
```

```
60      2 port1 bclr                        \ clear bit 2
61    ;
62
63    : Lad-an ( -- )                       \ Lader an
64      3 port1 bset                        \ set bit 3
65    ;
66
67    : Lad-aus ( -- )                      \ Lader aus
68      3 port1 bclr                        \ clear bit 3
69    ;
70
71    : Ubatt ( -- v )                      \ Messe Akku, Channel 6
72      6 adc@                              \ 400 entspricht 10,7 V
73    ;
74
75    : Messen ( -- v )                     \ Mittelwert der Akkuspannung
76      Des-aus Lad-aus &10 ms 0 Akku !     \ alles auf Null
77      &10 0 do                            \ Mittelwert bilden
78        Ubatt &10 / Akku +! 1 ms          \
79      loop                                \
80      Akku @                              \ und holen
81    ;
82
83    : Steuern ( v -- )                    \ Lader/Desulfator an/aus
84      &404 >                              \ Akku > 10,8 V ?
85      if Des-an                           \ ja, nur Desulfator an
86      else Tag?                           \ nein, ist Tag?
87        if Lad-an Des-an                  \ ja, dann beide an
88        else Lad-an then                  \ Resthelligkeit nutzen
89      then                                \ ... bis demnaechst
90    ;
91
92    : Not-aus                             \ Tiefentladung vermeiden
93      Ubatt &400 <                        \ Akku < 10,7 V ?
94      if Lad-aus then                     \ ja, alles aus
95    ;
96
97    : Dauerlauf ( -- )                    \ Doit
98      AdcInit                             \ initialise Adc
99      begin
100       Hell Messen Steuern Not-aus       \ Schalter setzen
101       10s key?                          \ warten und Watchdog blinken
102     until
103   ;
104   ' Dauerlauf is bootmessage            \ laufe los nach Reset bis Taste
105
106   ram
107   savesystem                            \ "empty" vorher nicht vergessen?
```

Programm 2 — Scope.fs

```
1   \ Scope.fs     Stand 2010-04-14
2   \ Scope mit R8C/13           rs
3
4   ram
5   : Doit ;                \ for MARKER
6
7   %00000010 constant PD0   \ PRC2-bit
8   $0A       constant PREG  \ Protect Register
9   %00000001 constant TxD1  \ bit 0 is Output, bits 1..7 are ADC-Inputs
10  $E2       constant PDIR  \ Direction Register port0
```

```
11  $E5       constant port3 \ P30..P33 Input (not yet used)
12
13  variable Chan
14  variable Flucht
15  create AdBuf &500 chars allot
16
17  rom
18
19  : AdcInit ( -- )          \ override lcdinit
20      PD0 PREG bset         \ enable PD0 in Protect Register
21      TxD1 PDIR c!          \ bit 0 is TxD1 Output,
22      &20 ms                \ bit 1..7 is Input Channel 6..0, settle
23  ;
24
25  : Channel ( -- )
26      begin
27        key dup Chan c! dup      \ Kanal 0..6 erlaubt
28        $20 = if
29          $20 Flucht !           \ space from HyperTerminal aborts
30        then
31        6 > while
32      repeat
33  ;
34
35  : Val@ ( chan -- val )
36      $80 + $D6 c!               \ fAD/2 selected (adcon0)
37      $20 $D7 c!                 \ Vref connected, 8 bit (adcon1)
38      6 $D6 bset                 \ start
39      begin
40        6 $D6 btst 0=            \ still converting?
41      until                      \ no, ready
42      $C0 c@                     \ get value
43  ;
44
45  : Sample ( chan -- )
46      &500 0 do dup              \ Kanal duplizieren
47        Val@ AdBuf i + c!        \ fill buffer
48      loop drop                  \ letztes Duplikat entfernen
49      &500 0 do
50        AdBuf i + c@ emit        \ send buffer
51      loop
52  ;
53
54  : Blink
55      port1 c@ $0F and 1 xor led!
56  ;
57
58  : Scope ( -- )
59      AdcInit 0 Flucht ! 0 led!  \ AD initialisieren, keine Flucht
60      begin
61        Channel Chan c@          \ Kanal bestimmen, an Sample geben
62        Sample Blink             \ messen, zeigen und blinken
63        $20 Flucht @ =           \ bis Abbruch, dann ^C und Space im HT
64      until begin key? until
65  ;
66  ' Scope is bootmessage         \ laufe los nach Reset bis Leertaste
67
68  ram
69  savesystem                     \ "empty" vorher nicht vergessen?
```

König Artus' Tafelrunde und König Minos

Martin Bitter

Mein Umgang mit xbigforth's GUI–Generator

Für Rechtschreibfans: Der englische König Arthur heißt im Deutschen meist Artus. Der Genitiv im Englischen wäre also Arthur's und im Deutschen Artus' oder Artusens (veraltet).

Die Tafelrunde

Begegnet ist mir die Aufgabenstellung auf der Wanderausstellung Mathematikum [1], die im Jahr der Mathematik den Menschen der Republik Mathematik anschaulich und faszinierend machen sollte. In eine runde Platte waren 7 oder 9 Leuchten mit Tastern eingelassen. Auf Tasterdruck änderten jeweils drei Leuchten ihren Zustand zwischen an und aus. Die Geschichte um König Artus habe ich mir als (hoffentlich) Bereicherung ausgedacht.

Bekannt ist Artus' Tafelrunde. Dadurch, dass die Tafel (der Tisch)[1] rund war, gab es keinen hervorgehobenen Sitzplatz — alle waren von der Sitzordnung her gleich. Der König (Artus) selbst speiste an einem anderen Tisch, um die Ordnung der Gleichen nicht zu stören. Die Anzahl der Ritter, die an dem Tisch saßen, schwankte im Lauf der Zeit und je nach Überlieferung zwischen 12 und 16 (siehe Wikipedia [2]). Hier in meinem Programm können es vier bis 12 Ritter sein. Die Ritter sind bereit, sich an einen rechteckigen[2] Tisch zu setzen. Dabei haben sie eine recht verschrobene Art, Platz zu nehmen. Sobald ein Ritter Platz nimmt oder ihn verlässt, hat dies Auswirkungen auf die benachbarten Sitze. Ist ein Nachbarsitz unbesetzt, so setzt sich dort ein weiterer Ritter hin. Ist der Nachbarsitz allerdings schon besetzt, so wird der dort sitzende Ritter ihn räumen. Es können bei drei nebeneinander stehenden freien Stühlen sich drei Ritter gleichzeitig setzen. Andererseits kann ein sich setzender Ritter zwei Ritter von ihren Stühlen drängen. Genau das Gleiche gilt, wenn ein Ritter seinen Sitzplatz räumt. Sind die Nachbarsitze besetzt, so werden die betreffenden Ritter sie verlassen. Sind sie leer, so werden sich dort Ritter niederlassen. Etwas mathematischer und klarer ausgedrückt: Jeder Stuhl kann zwei Zustände haben: leer (`false`) oder besetzt (`true`). Wird der Zustand eines Stuhles geändert (durch Räumen oder Besetzen), so ändern sich die Zustände der beiden benachbarten Stühle.

Wie oft nimmt ein Ritter Platz, wenn die Tafel für vier, fünf, ... 12 Ritter gedeckt ist? Für sechs, neun und 12 Plätze ist die Lösung einfach: jeder setzt sich einmal hin — wenn man es klug anstellt.

Die anderen Fälle kann man mit dem Programm `KingArthur` ausprobieren. Eine ANS-94–Version für bigforth oder gforth ist schnell geschrieben (siehe Listing 1). Die Klammern in Zeile 2 und 7 für gforth bitte entfernen[3]. Mit Eingabe einer Ziffer wird der entsprechende Platz in der unteren Reihe be– oder ent–setzt. Die Nachbarn folgen der Regel. Bedenken muss man nur eines: eine runde Tafel wird hier auf eine gerade Anordnung projiziert. D. h. die beiden Eckplätze beeinflussen

[1] das Wort Tisch wiederum kommt von dem griechischen Diskus, und bezeichnet eine runde (vormals Baum–) Scheibe. Eine Tafel ist, wie heute noch in vielen Schulen zu sehen, rechteckig. *Tafelrunde* wörtlich genommen ist also ein Widerspruch in sich selbst.

[2] dies hat programmtechnische Gründe

[3] Die Klammern in Zeile 7 bügeln ein unerwartetes Verhalten von `mod` in gforth wieder glatt.

```
1  \ Ein kleines Logikspiel                          23aug09mb
2  ( : at at-xy ; )
3  Variable Anzahl              8 Anzahl !
4  4 Value mindestens           9 Value höchstens
5  Create Leuchten höchstens allot
6  : schalten ( n -- )
7    ( Anzahl @ + ) Anzahl @ mod Leuchten + dup c@
8    IF false ELSE true THEN swap c! ;
9  : 3schalten ( n -- )
10   dup 1+ schalten dup 1- schalten schalten ;
11 : zeigen ( -- ) 0 0 at Anzahl @ dup
12   cr 0 DO I 3 .r 2 spaces LOOP
13   cr 0 DO Leuchten I + c@ 3 .r 2 spaces LOOP ;
14 : start page Leuchten höchstens erase zeigen
15   BEGIN 1 pad c! key pad 1+ c! pad number?
16   WHILE 3schalten zeigen REPEAT drop ;
```

Listing 1: `KingArthur`: König Artus' Tafelrunde als Forth–Konsolenprogramm

```
0   1   2   3   4   5   6   7
0 255 255   0 255 255   0   0
```

Screenshot 1: `KingArthur` nach dem Drücken der Tasten `2` und `4`

sich gegenseitig (der *Kreis* ist ja genau hier geschlossen). Vergleiche Screenshot 1: Hier wurde zuerst Taste 2 und dann die Taste 4 gedrückt.

Es gibt eine kluge Lösungsstrategie, die die Ritter anwenden können. Bei vier Rittern setze man sich in folgender Reihenfolge auf die Stühle: `1`, `3`, `2` und `4`. Fertig!

Bei Hofe

Wirklich fertig? In der Überschrift ist von zwei Königen die Rede! Artus und ... Minos! Minos — wer ist denn das? Ein uralt kretischer König, ein Bezwinger der Meere und Nennvater des Minotaurus, den zwar Minos' Frau gebar, dessen Vater er (Minos) aber nicht war. Und ... und ... und.

Minos nun kann mir helfen, das gleiche Problem, das mit einer kargen Textausgabe auf einem Screen gelöst werden konnte, mit mehr Aufwand und komplexeren Befehlsketten mit einem schönen GUI zu versehen. Minos war nämlich nicht nur der besagte König, Minos ist auch eine Forth–Bibliothek — in xbigforth — die eine Unmenge an grafischen Elementen zur Verfügung stellt, um *schöne* bunte Bilderoberflächen zu gestalten.

Buttons, Sliders, Radioknöpfe, Textfelder, Menus — alles was man für ein *schönes* GUI benötigt, ist vorhanden. Ja sogar fertige Leinwände und ein Editor gehören dazu. All' diese grafischen Elemente sind Forthwords, genauer Methoden. Sie lassen sich wie jedes word aufrufen. Parameter werden auf dem Stack übergeben. Man kann den Code für ein solches GUI word für word, Methode für Methode schreiben — oder einen Gutteil der Arbeit dem GUI–Editor überlassen, der bei bigforth mitgeliefert wird.

Nicht malen — denken!

Dieser GUI–Editor heißt Theseus und ist es wert, oft verwendet zu werden. Diejenigen, die ihn kennen, verwenden ihn gerne. Er führt dennoch (teilt er dies mit Forth?) das Dasein eines Mauerblümchens. Der vorliegende Artikel soll dazu ermuntern, ihn einmal auszuprobieren.

Im Gegensatz zu anderen GUI–Editoren, die ich kenne (alte Erinnerungen aus der Windowswelt), verfolgt Theseus kein malendes, sondern ein logisches Konzept. In der Windowswelt erklickte ich einen Button, fasste und positionierte ihn dann mit der Maus. Beim Einfügen weiterer Buttons musste ich dann gelegentlich den ersten Button verschieben ... gelegentlich sogar oft.

Bei Theseus ist dies anders — und nach der Eingewöhnungszeit empfindet man dies sogar als Erleichterung. Hier legt man nicht die Position einzelner Elemente fest, sondern die logische Anordnung. Es heißt nicht Button1 hierhin und Button2 dahin, sondern Button1 und Button2 neben(über)einander, den *Rest* erledigt Theseus für mich. Ich muss mich zwingen, vor dem Basteln des GUIs die Logik — einigermaßen — zu überlegen. Daraus folgt eine Beschränkung: Ich kann (noch)[4] nicht die Lage einzelner Elemente frei bestimmen.

Schritt für Schritt

Als Appetitmacher dient mir König Artus' Tafelrunde. In einem ersten Teil, soll dieses Programm mit einem

[4] Ewald Rieger soll dazu in der Lage sein, „... aber für eine Einführung ist das zu komplex."

[5] Hierzu sei die Lektüre der Dokumentation zu bigforth empfohlen, die ebenfalls bei Bernds Site zu bekommen ist. s.o.

Über bigforth ...

bigforth wurde von Bernd Pasysan in den 1990ern geschrieben und seitdem weiter gepflegt. Es läuft unter Linux und Windows (mit Einschränkungen). bigforth ist ein reines Konsolenprogramm, d. h. ohne graphischen Schnick–schnack. Aber es bringt einen Bruder (eine Erweiterung) mit sich: xbigforth. Ein bigforth, das unter und mit dem graphischen Fenstersystenm X–Windows funktioniert.

bigforth kann jederzeit von Bernd Paysans Seite [3] heruntergeladen werden. Linux–Laien sei die Debian–Version empfohlen. Jene, die mit Linux und SVN vertraut sind, werden sich stets auf die aktuellste Version berufen können. Andererseits können auch Laien wie ich in den Genuss der SVN–Version kommen: (das Paket `subversion` muss installiert sein)

```
svn co http://www.forth-ev.de/repos/bigforth
```

danach in das frisch erzeugte (oder überschriebene) Verzeichnis bigforth wechseln und

```
autoconf
./configure
make install
```

eingeben.

In der Datei `/usr/local/lib/bigforth/.xbigforthrc` bzw. `.bigforthrc` können jetzt Suchpfade und persönliche Einstellungen geändert werden.

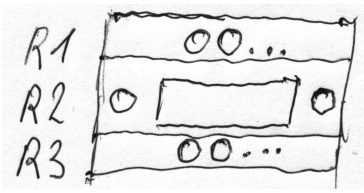
Fallstricke

Die Einstellungsdatei `.xbigforthrc` ist eine ganz normale Forthquelle, die interpretiert wird. Falls eine Eingabe zu einem Fehler führt, wird nicht nur forthtypisch an der Stelle des Fehlers abgebrochen, sondern der geänderte Pfad wird verworfen und der in der .bigforth angegebene Pfad verwendet. bigforth unterscheidet seine Quelldateien anhand der Endungen: `.f` bedeutet: altes Screenformat 16 Zeilen a 64 Zeichen **ohne** LF oder CR Zeichen. `.fs` steht für ein (kürzeres) Fließtextformat mit CR und LF. Beim Laden werden zwischen den Formaten Unterschiede gemacht. `*.f`–Dateien werden über *alte* Blockbefehle (`make use load`) angesprochen — `*.fs`–Dateien über das modernere `include`: Also bei der Verwendung externer Editoren darauf achten, wie man seine Entwürfe benennt. Tippfehler verzeiht bigforth im Allgemeinen. Groß–Klein–Schreibung spielt (default) keine Rolle. Außer beim Laden von Dateien: `Include Dateix.fs` legt zuerst ein internes word `Dateix.fs` an, sucht dann diese Datei in den Suchpfaden und lädt sie. Passiert hier ein Tippfehler z. B., `include dateix.fs`, wird ein word `dateix.fs` erzeugt und die entsprechende Datei nicht gefunden (die heißt ja `Dateix.fs` mit großem `D`). Selbst ein nachfolgendes richtiges `include Dateix.fs` führt nun zu einer Fehlermeldung, weil das interne word `dateix.fs` angesprochen wird, das ja ins *Leere* zeigt. Abhilfe: Neustart.
Natürlich ist es legitim, die von Theseus erzeugten `*.m`–Dateien mit einem externen Editor (im Textformat) zu bearbeiten. Manchmal ist das einfach schneller und bequemer (z. B. Pfadangaben bei Icons ändern). Aber Vorsicht! Theseues öffnet eine solche Datei nicht nur, sondern er lädt sie, d. h., sie wird interpretiert. Das Ergebnis dieser Interpretation wird im Theseusfenster angezeigt und eben **nur** dieses Ergebnis wird beim Speichern auch gesichert. Fügt man also Kommentar**zeilen** ein, so sind diese verloren. Schlimmer noch, hat man einen (vorerst unsichtbaren) Fehler provoziert, so findet man nach dem Abspeichern nur noch eine fast leere Rumpfdatei vor. Also: Immer Sicherheitskopien machen — oder keine externen Editoren verwenden. Manchmal kann es vorkommen, dass die Eingabemaske des Inspectors größer ist als der angezeigte Bereich. Ich habe so manchmal übersehen, dass noch einiges eingegeben werden musste. Also immer schön darauf achten, ob rechts im Inspector ein Slider auftaucht.

GUI versehen werden und erst einmal nur laufen. In einem zweiten kann es durch GUI–Standardelemente (Menu) erweitert werden. Ich möchte nicht den Umgang mit Theseus erklären[5], sondern zeigen, wie ich mit ihm arbeite — vielleicht kann ja der eine oder andere, der mir Schritt für Schritt folgt, daraus für sich einen Gewinn ziehen. Wie immer stehen die Quelldateien zum Download bereit. Zu jedem Schritt gibt es je eine Datei `ka1.m`, `ka2.m`, `ka3.m` etc. Wer mag, kann diese laden und testen, wenn sie im folgenden Artikel erwähnt wird.

Präludium

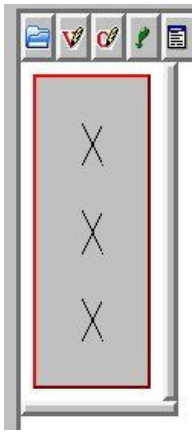
Vorüberlegungen: Ich stelle die Tafelrunde als Grundriss dar. Gerne würde ich die Sitzplätze in einem Kreis anordnen, aber Theseus ist rechtwinklig angelegt. Also setzte ich die *Ritter* an einen rechteckigen Tisch. Weiter: Ich möchte die Kopfseiten des Tisches besetzen, also benötige ich in meinem Grundriss drei Reihen: je eine Reihe mit Sitzplätzen an jeder Längsseite und den Tisch selbst mit den Sitzplätzen an den Kopfseiten.



Ich könnte pro Aufgabenstellung (vier, fünf, ... 12 Sitzplätze) ein je eigenes Spielfeld (Dialog) basteln und dieses je nach Fall laden ... zu viel Arbeit. Also benötige ich eine Methode, Sitzplätze zu einem bestehenden Spielfeld hinzuzufügen oder zu entfernen. Da weiß ich (noch) nicht, wie das geht.

Möge die Übung gelingen!

xbigforth starten und im Menu `File` das Item `Theseus` anklicken. Im nun auftauchenden Fenster (siehe Doku zu bigforth) einmal (oben rechts) auf `vbox` klicken. Im Dialogfenster erscheint nun ein rot hervorgehobenes, durchge–x–tes Quadrat, dem wir den Namen (`Name:`) `Spielfeld` geben. Als Titel erhält es: (`Title:`) `König Artus' Tafelrunde`. Ich habe eine `vbox` (v=vertical) gewählt, weil ich innerhalb dieser box drei weitere Boxen senkrecht untereinander anordnen will. Das ist mit drei schnellen Klicks auf `hbox` erledigt. `hbox` (h=horizontal) wähle ich, weil in den drei Reihen die Elemente Tisch bzw. Stühle nebeneinander angeordnet werden sollen. Jetzt aktiviere ich durch Anklicken die obere der drei hboxen, und ... mhm, wie stelle ich jetzt einen leeren Sitzplatz dar?



Genaues Ansehen der Menus zeigt mir, dass einige Elemente mit Icons (das sind kleine Bilder) verbunden werden können: Icons eignen sich gut, *versteckte* Buttons anzulegen. Darunter verstehe ich Buttons, die nicht 3D–artig hervorgehoben werden, sondern sich vom Hintergrund nicht abheben. Im Menu `Buttons` gibt es u. a. `Icon-Button`, `Icon`, `Big-Icon` und im Menu `Toggles` gibt es: `Icon`, `Flip-Icon` und `Iconbutton`. Ich probiere mal alle der Reihe nach aus. Draufgeklickt und das Ergebnis gesichtet. Es bleiben zwei Möglichkeiten für *versteckte* Buttons offen: Unter den `Buttons` ist es `Big-Icon` und bei den `Toggles` ist es: `Iconbutton`. Beide bieten nicht nur

Der Classbrowser

Die Minos–Elemente sind Forthobjekte. Um sie, ihre Abhängigkeiten und Hierarchien, zu erkunden, kann man das altbekannte `words` benutzen. So listet die Eingabe `Spielfeld words` nicht nur die Unterobjekte und Funktionen von Spielfeld auf, sondern zeigt auch an, welche von *außen* erreichbar sind (public) und welche nicht (private). Einen schnelleren und informativeren Überblick bietet der Classbrowser im Menu `File` von xbigforth. Dort werden alle zur Zeit geladenen Methoden in einem hierarchischen Baumgraphen angezeigt und farblich gekennzeichnet (wobei ich noch nicht ganz durchschaue, was die Farben bedeuten).

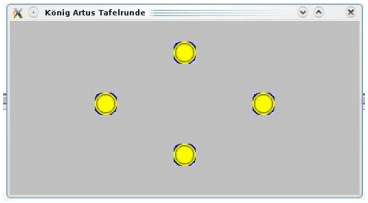
Platz für ein Icon, sondern auch für einen String, (die Beschriftung des Buttons). Jetzt benutze ich die Möglichkeit, die Elemente, die ich durch wildes Herumklicken erzeugt habe, wieder zu entfernen: Indem ich bei gedrückter Groß–Taste (Shift) ein Element mit der linken Maustaste anklicke, wird es aus dem Dialog entfernt. (Es ist nicht wirklich gelöscht, sondern liegt auf einem Objektstack, von dem es mit Groß–Taste+rechtem Mausklick wieder abgerufen werden kann. Eine copy–Funktion für Elemente gibt es nicht.) Wer jetzt neugierig ist, kann das Projekt schon mal speichern. Entweder über das Menu `File` oder über eines der kleinen Symbole in der linken Symbolleiste (mit der Maus darüber fahren, die aufpoppenden Texte erklären die Funktion). Wichtig ist es, nicht die Endung `.m` zu vergessen. Ich wähle hier als Dateinamen `ka0.m`. Das kann als **K**eine **A**hnung, oder als **K**ing **A**rthur gelesen werden.

First Sight

Im xbigforth–Fenster jetzt ein `include ka0.m` eingegeben und (hoffentlich) ein `ok` erhalten? Dann kann es weiter gehen mit `Spielfeld open-app`. Auf dem Bildschirm erscheint jetzt unser Spielfeld so, wie wir es von Theseus her kennen. Ein bisschen damit spielen: vergrößern, verkleinern, ziehen und bewegen geht, die Max–Min–Schließ–Felder tun es auch. Ich kann mir mit dem Classbrowser (siehe auch Kasten *Der Classbrowser*) die internen Möglichkeiten von Spielfeld anschauen. :-)

Stuhlgang?

Jetzt weiter: Wieder Theseus gestartet und die Datei `ka0.m` geladen. Meine Überlegung ist ja, die Plätze (Ritter) um einen rechteckigen Tisch zu platzieren. Bei vier Rittern ist das: einer oben, einer rechts, einer links und einer unten.



Dazu klicke ich in die obere der drei hboxen (sie wird rot umrandet). Jetzt wähle ich aus dem Menu `Buttons` ein `Big-Icon`. In dem unteren Bereich des Theseus–Fensters öffnet sich eine Eingabemaske (Inspector genannt) in der ich dem neuen Objekt aus der Klasse Big–Icon einen Namen (`Stuhl1`) gebe. Das ist wichtig, weil ich dieses Objekt später nur (?) über seinen Namen ansprechen kann. Die Buttonbeschriftung brauche ich nicht, also lösche ich im Inspector jeden Inhalt, der zum Eingabefeld `String:` gehört. Jetzt muss ich noch einen (zu meinen Suchpfaden in `.xbigforthrc` passenden) Pfad angeben, der auf eine Bilddatei im `.png`–Format zeigt. Ich entscheide mich für `icons/piece0`[6], eines der vielen Bilder, die im bigforth Paket enthalten sind.

Nun die zweite hbox anklicken und genauso wie oben zwei Plätze einfügen (`Stuhl4` und `Stuhl2`). Dann in der unteren hbox wieder einen Platz (`Stuhl3`) einfügen. (Speichern (`ka1.m`) und in bigforth laden: `include ka1.m` und dann `Spielfeld open-app`).

Bitte setzen!

Das muss schöner werden, aber zuerst will ich *spielen* (siehe Kasten *Spielen*). Wie kann ich die Icons ändern? Ein Blick in die `ka1.m`–Datei zeigt mir die Wordsequenz, mit der die Big–Icons erzeugt werden[7]. Offenbar legt das word `icon"` einen Stack für die

[6] Wenn ich keinen Pfad angebe, sucht Minos die `.png`–Datei im Ordner `icons`

[7] Auch der Classbrowser und Spielfeld words, sowie die Quelltexte sind hilfreich.

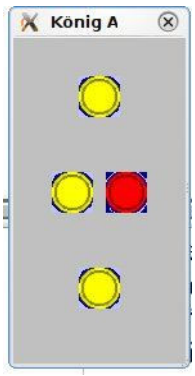
Spielen

Die `.m`–Datei erzeugt eine Klasse Spielfeld, die ich von außen nur bedingt ansprechen kann. Nachdem Laden von `ka0.m` kann ich zwar `Spielfeld words` eingeben, aber `Spielfeld Stuhl1 words` führt zu einem Busfehler. Will ich *spielen*, so muss ich mir erst einmal ein Object der Klasse `Spielfeld` erzeugen, auf das ich dann zugreifen kann. Das macht die Sequenz:

```
Spielfeld ptr Spiel
Spielfeld new bind Spiel
```

mit `Spiel self 0 s" Spielwiese" open-component` öffne ich nun dieses Object und kann spielen. `Spiel Stuhl1 words` zeigt mir jetzt die Methoden, die `Stuhl1` beherrscht. Ich mache nun die Minos–Worte mit `minos also` sichtbar. (Vorsicht beim *Spielen*, es befriedigt zwar die schnelle Neugier, aber nicht alle Worte, die man damit erzeugt, sind später innerhalb der `.m`–Datei verwendbar!)

Icons an, übergeben wird letztendlich eine Adresse. Leider führt die Eingabe von `icons" icons/piece0"` zu einem `don't know icon` bzw. `compile only icon` — aber ich kann es in einer Definition verwenden: `: platz ( -- adr ) icon" icons/piece1"`.



Mithilfe von `Spielfeld words` bzw. `Spiel words` sehe ich, dass mein Objekt `Spielfeld` über die Unterobjekte `Stuhl1` bis `Stuhl4` verfügt. Ich weiß, dass diese Big–Icons sind. Über diese verrät mir der Classbrowser bzw. `Spiel Stuhl1 words`, dass sie eine Methode `assign` haben. Jetzt wird's spannend. Ich gebe ein `platz Spiel Stuhl1 assign` und das gelbe Icon wird durch ein rotes ersetzt. Ha: ich bin König! Ich kann was! Nämlich Icons ändern. Nun baue ich mir Bilder im `.png`–Format mit Gimp (oder einem x–beliebigen Zeichenprogramm) und mache sie mir leicht verfügbar mit

```
: leer ( -- adr ) icon" leer.png" ;
```

und

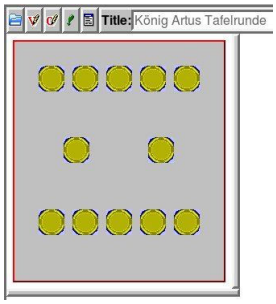
```
: voll (-- adr ) icon" voll.png" ;
```

. Das Wort

```
: Sitz1 ( adr -- ) spiel Stuhl1 assign ;
```

erlaubt mir dann im xbigforth–Fenster solche Dinge wie `voll Sitz1` bzw. `leer Sitz1`[8]. Da ich die Objekte `Stuhl1`, `Stuhl2`, `Stuhl3` und `Stuhl4` (noch) nur über ihre Namen ansprechen kann, muss ich mir für jeden Platz ein solches Wort schreiben. Also definiere ich mir analog zu `Sitz1` die Worte `Sitz2`, `Sitz3` und `Sitz4`.

Sitzordnungen



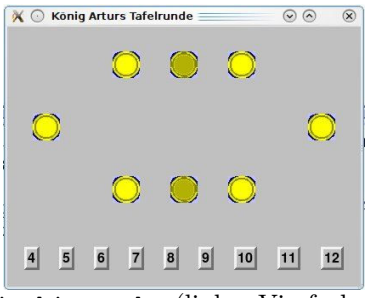
Genug gespielt! Es geht ernsthaft weiter mit Theseus/Minos. Wie schon erwähnt kann ich (noch?) nicht im laufenden Programm neue Big–Icons in den Spielfeld–Dialog einfügen bzw. entfernen. Also entwerfe ich (mit Gimp) eine dritte `.png`–Datei in der Größe der beiden Icons `leer.png` und `voll.png`, die nur einen transparenten Hintergrund enthält, und nenne sie `weg.png`. Nun öffne ich Theseus, lade `ka1.m`, aktiviere durch Anklicken das erste Big–Icon (`Stuhl1`) und klicke 4–mal darauf. Es werden vier neue Big–Icons in einer Reihe nebeneinander erzeugt. Jedes versehe ich mit der `voll.png` und lösche den Inhalt des Textfeldes. Ähnlich verfahre ich mit dem untersten Big–Icon (`Stuhl3`). Nun habe ich zwölf Big–Icons, die die zwölf möglichen Stühle in der Tafelrunde darstellen. Ich aktiviere jedes einzeln im Uhrzeigersinn und benenne es neu: `Stuhl1` bis `Stuhl12` (das hilft mir, später den Überblick zu bewahren) (`ka2.m`).

Zufriedenstellend sieht das ja nicht aus. Die Position der Icons ist durch Minos festgelegt — aber nach meinen Vorgaben. Wie sag ich ihm, was ich will? In diesem Fall helfen die glue–Objekte. Obwohl *to glue* kleben bedeutet, sind die glue–Objekte kein reiner *Festkleber* sondern auch Verdränger. Da ich im Nachhinein etwas einfüge, ist es wichtig darauf zu achten, wohin meine neuen Objekte platziert werden. Darüber entscheiden die vier Buttons links im Theseus–Fenster und das darunterliegende Pfeil–Feld. Ich wähle: `Add after current Objekt`. Dann öffne ich das `Glues`–Menu, klicke auf den ersten Platz (aktiviere `Stuhl1`) und klicke auf das Item `Glue`. Dadurch wird ein Glue–Object rechts des jeweiligen Big–Icons eingefügt. Genauso verfahre ich mit `Stuhl2`, `Stuhl3`, `Stuhl4` und mit `Stuhl11`, `Stuhl10`, `Stuhl9`, `Stuhl8`. Bei all diesen ändere ich an der Standardeinstellung der Glues im Inspector nichts, d. h., ich lasse sie auf `fill`. Die Big–Icons `Stuhl5`, `Stuhl6` und `Stuhl7` erhalten keinen Glue an ihrer rechten Seite. Zum Schluss aktiviere ich `Stuhl12` und füge dort ein weiteres Glue–Objekt ein, dem ich diesmal im Inspector die Eigenschaft `fillll` zuweise. Dieser (oder dieses?) Glue drängt nun die beiden Stühle `Stuhl6` und `Stuhl12` weit nach außen. Abstände berechnen oder gar Pixel zählen muss ich nicht! Mit diesem Ergebnis kann ich leben (`ka3.m`). (Wer mir bis hierher gefolgt ist, kann den Glue zwischen `Stuhl12` und `Stuhl6` durch ein Rechteck ersetzen, das den Tisch darstellt.)

Was geht?

Zwischenstand: Ich habe eine Anordnung von Big–Icons (Stühlen), die mir einigermaßen gefällt. Ich kann jedem `Stuhl` drei Zustände zuweisen: besetzt (voll), unbesetzt (leer) und verschwunden (weg).

Weg da!



Nun will ich dem Spieler die Möglichkeit geben, die Anzahl der Stühle zu variieren. Das kann später über ein Menu geschehen, hier und jetzt löse ich das über Buttons. Bei geöffnetem `ka3.m` klicke ich solange auf `Up in hierarchy` (links, Vierfachpfeil), bis ich in der obersten Ebene bin, wähle (links) `add after last in box` und füge eine vierte hbox ein. In diese füge ich (aus dem Menu `Buttons`) einen einfachen Button ein. Einen Namen gebe ich ihm nicht, da ich ihn im Programm nicht ansprechen muss. Wie schon bei den Stühlen beschrieben, füge ich acht weitere Buttons

[8] Man beachte die Tücken des Spielens! (siehe Kasten *Fallstricke*)

und dazwischenliegende Glues ein. In die Textfelder der Buttons trage ich die Zahlen 4 bis 12 ein (`ka4.m`).

. . . und Action!

Optisch bin ich jetzt fertig — es fehlen noch die Aktionen, die durch Klicken auf die Stühle (Big–Icon) und Zahlen (button) ausgelöst werden. Jetzt kommen 5 weitere Symbole links unter der Objektleiste der Theseus–Oberfläche ins Spiel: `Dialog Editor`, `Edit Variables`, `Edit Methods`, `Show Dialog` und `Dialog Menu` (die Namen werden bei Mauskontakt angezeigt). Ich benutze davon `Edit Variables`, `Edit Methods` und `Dialog Menu`. Nach einem Klick auf `Edit Variables`, gebe ich `Variable leds#` (hier soll die Anzahl der jeweils aktiven Stühle gespeichert werden) ein. Damit wird eine Variable `leds#` erzeugt, die zum Objekt `Spielfeld` gehört. Davon kann ich mich leicht überzeugen, indem ich die `ka5.m`–Datei in xbigforth lade (`include ka5.m`) und `Spielfeld words` eintippe. Dies zeigt mir, dass diese Variable im Bereich `public` liegt, das heißt, sie kann von *außen* z. B. mit Spielfeld `leds# @` angesprochen werden.

Unter `Edit Methods` öffnet sich ein Texteditorfeld in dem ich — wie im ganz normalen Forth — words definieren kann. Diese words sind Methoden des jeweiligen Dialog(objektes) im *privaten* Bereich.

Unter den Items des kleinen `Dialog`–Menus klicke ich auf `Edit impl`. Theseus erzeugt nun aus dem Namen meines Dialoges (`Spielfeld`) einen Dateinamen `Spielfeld.fs` und versucht, eine solche Datei zu öffnen, findet sie nicht und beschwert sich (in der xbigforth Konsole) darüber. Nachdem ich diese Datei mit normalen Mitteln (außerhalb) angelegt habe, wird sie gefunden und in einem Editor geöffnet. An diesen Namensvorschlag muss ich mich nicht halten, in dem Textfeld hinter `Edit impl` kann ich bei Bedarf Pfad– und Dateinamen ändern. In dieser Datei können wiederum alle Forthbefehle des xbigforth–Systems benutzt werden. Definitionen werden in der aktuellen Wortliste abgelegt und sind ohne Objekt–Methodenbedingte Einschränkungen nutzbar. Ich definiere mir einige allgemeine Dinge in `Spielfeld.fs` (siehe auch Listing 2 auf der nächsten Seite): die Höchstanzahl der darzustellenden Stühle, Icons für einen nicht vorhandenen, einen besetzten und einen leeren Stuhl, ein Array, in dem alle möglichen Sitzordnungen als Bitfelder gespeichert sind, und zwei Worte, die die Stühle zwischen besetzt (`ein`) und unbesetzt (`aus`) wechseln. Das Wort `Zustand_schalten` schaut zuerst nach, ob der Stuhl an der betreffenden Position überhaupt vorhanden ist ($\neq 0$), und schaltet nur zwischen den Zuständen `1` und `2` um. Dabei hinterlegt es ein Flag, das ein erfolgreiches Umschalten anzeigt. Das Wort `umschalten` benutzt eben dieses Flag, um zu entscheiden, welche benachbarten Stühle ihren Zustand ändern müssen.

Klick mich!

Ein Programm reagiert meist auf innere oder äußere Ereignisse. Oft sind dies Benutzereingaben. Ein Programm mit einem GUI soll auf Mausklicks reagieren. Hier liegt nun eine Stärke von Minos/Theseus. Im Theseus–Editor kann ich zu jedem Element mit normalen Forth–Worten festlegen, auf welche Ereignisse und wie das jeweilige Element reagieren soll.

Dazu habe ich unter `Edit-Methods` (siehe oben) folgende Worte definiert:

`led ( icon Stuhl - ) \`
: weist dem Icon/Stuhl mit der Nummern n ein Bild aus dem Iconstack zu

`schalten ( Zustand Stuhl - ) \`
: weist dem Icon mit der Nummer n abhängig vom Zustand n ein Bild zu.

`leds ( n - ) \ Stuhlbild`
: liest das zur Anzahl der Stühle passende Stuhlbild aus und erzeugt die entsprechenden Zustände für die einzelnen Stühle.

`anzeigen ( - ) \ Stühle zeichnen`
: zeichnet alle Stühle, auch die *unsichtbaren* (besetzt, frei, unsichtbar)

`Spielzug ( n - ) \`
: ändert betroffene Stühle und zeichnet die gesamte Sitzordnung neu.

Die Nummern–Buttons auf dem Spielfeld (4 bis 12) sollen die jeweilige Stuhlanordnung zeichnen.

Deshalb aktiviere ich im Theseus–Editor den Button 4 und gebe in seinem Codefeld (unten im Inspector) `4 leds` ein. Entsprechend verfahre ich mit den Buttons 5 bis 12. Das war's! Nun führen die Nummern–Buttons beim Anklicken die gewünschte Aktion aus. (`ka5.m` Zeile 30 bis 62)

Als (vorläufig) Letztes bleibt nun noch, den *Stühlen* ihre Aktion zuzuweisen. Der Reihe nach aktiviere ich `Stuhl1` bis `Stuhl12` und gebe in ihrem Codefeld (siehe Inspector) `0 Spielzug`, `1 Spielzug` etc. ein. (`ka6.m` Zeile 65 bis 83).

Fertig!

Jetzt kann im geöffneten xbigforth–Konsolenfenster mittels `include ka6.m main` das fertige Spiel geladen und gespielt werden. Viel Spaß!

Ausblick

Falls es zu einem Nachfolgeartikel kommt, zeige ich, wie ich ein Menu gestalte und wie ich es schaffe, die Stühle in einem Kreis aufzustellen. Schreibt mir, ob Ihr Interesse an einer Fortsetzung habt.

Nachwort

Bernd Paysan ist einer der ersten Leser dieses Artikels. Er sandte mir seinen Vorschlag, bei dem er die Einstellungen der Stühlezahl über ein Sliderobjekt gelöst hat. Damit der geneigte Leser dies nachvollziehen kann, füge ich die Datei `ka7.m` zum Download bei.

```
12 Constant Max#                \ Maximalanzahl der Stühle

: aus icon" LED_ge_aus.png" ;   \ zur bequemeren Verwendung, drei Wörter,
: ein icon" LED_ge_an.png" ;    \ die die Adressen der jeweiligen Icons
: weg icon" LED_ge_weg.png" ;   \ aus dem Iconstack holen.


Create Spielfelder              \ Bitfelder, die die hübschen? Sitzanordnungen enthalten
  %100100100100 , \ 4-er
  %100100101010 , \ 5-er
  %101010101010 , \ 6-er
  %101110101010 , \ 7-er
  %101110101110 , \ 8-er
  %101110111011 , \ 9-er
  %111011111011 , \ 10-er
  %111111111011 , \ 11-er
  %111111111111 , \ 12-er

Create Zustand Max# allot       \ Speicher für an-aus, je ein Byte pro Stuhl


 : Zustand_schalten ( n -- flag )
  \ ändert (je nachdem) den Zustand des Stuhles Nummer n, zeigt an, ob etwas geändert wurde
  Max# mod zustand + dup                  \ Nummer in Adresse umrechnen, Kopie anlegen
  c@ 0 <>  tuck                           \ Zustand lesen, auf nicht-Null testen, Flag kopieren ( -- flag addr flag )
  IF dup c@ 1 = IF 2 ELSE 1 THEN swap c!  \ falls ja: Adr. kopieren, lesen auf 1 testen
                                          \ und entweder 1 oder 2 schreiben ( -- flag )
  ELSE drop                               \ falls nicht: Adresse vernichten ( -- flag )
  THEN ;

 : umschalten ( n -- )          \ LED-Nummer n und ihre Nachbarn ändern
  dup dup                       \ Nummer kopieren ( 3-fach)
  Zustand_schalten              \ versuche Zustand zu ändern (evtll.)
  IF                            \ falls etwas geändert wurde,
    BEGIN 1+ dup Zustand_schalten UNTIL drop    \ versuche Nachbar im Uhrzeigersinn umzuschalten, bis Erfolg!
    BEGIN 1- dup Zustand_schalten UNTIL drop    \ versuche Nachbar im Gegenuhrzeigersinn umzuschalten, bis Erfolg!
  THEN ;                        \ fertig
```

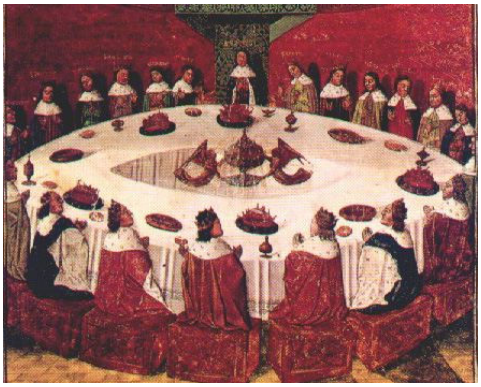
Listing 2: `Spielfeld.fs`

Links

[1] Wanderausstellung Mathematikum: `http://www.mathematikum-unterwegs.de/`
[2[Wikipedia zu Tafelrunde: `http://de.wikipedia.org/wiki/Tafelrunde`
[3] Bernd Paysans Bigforth–Seite: `http://www.jwdt.com/~paysan/bigforth.html`

Einladungen

Über die Menuepunkte `edit impl` bzw. `edit decl` wird die Reihenfolge, in der Dateien (nach)geladen werden, festgelegt. Eine `.fs`–Datei kann jederzeit eine `.m`–Datei laden. Ebenso kann eine `.m`–Datei eine `.fs`–Datei laden. Dies entweder nachdem sie ihre Definitionen bzw. Zeiger darauf kompiliert oder bevor sie selbst irgendetwas definiert. Dies hat Auswirkungen darüber, welche Programmteile wie auf die jeweiligen Worte zugreifen können.



Gral in der Mitte von Artus' Tafelrunde, französische Handschrift des 14. Jhs
Quelle: Wikimedia.org

Programm 0 — ka0.m

```
#! /usr/local/bin/xbigforth
\ automatic generated code
\ do not edit

also editor also minos also forth

component class Spielfeld
public:
 ( [varstart] )  ( [varend] )
how:
  : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
class;

Spielfeld implements
 ( [methodstart] )  ( [methodend] )
  : widget  ( [dumpstart] )
            cross new ( this is a stub )
          #1 habox new panel
            cross new ( this is a stub )
          #1 habox new panel
            cross new ( this is a stub )
          #1 habox new panel
        #3 vabox new panel
      #1 vabox new panel
    ( [dumpend] ) ;
class;

: main
  Spielfeld open-app
  event-loop bye ;
script? [IF]  main  [THEN]
previous previous previous
```

Programm 1 — ka1.m

```
#! /usr/local/bin/xbigforth
\ automatic generated code
\ do not edit

also editor also minos also forth

component class Spielfeld
public:
  big-icon ptr Stuhl1
  big-icon ptr Stuhl4
  big-icon ptr Stuhl2
  big-icon ptr Stuhl3
 ( [varstart] )  ( [varend] )
how:
  : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
class;

Spielfeld implements
 ( [methodstart] )  ( [methodend] )
  : widget  ( [dumpstart] )
            ^^ S[  ]S ( MINOS )  icon" icons/piece0" X" " big-icon new  ^^bind Stuhl1
          #1 habox new panel
            ^^ S[  ]S ( MINOS )  icon" icons/piece0" X" " big-icon new  ^^bind Stuhl4
            ^^ S[  ]S ( MINOS )  icon" icons/piece0" X" " big-icon new  ^^bind Stuhl2
          #2 habox new panel
            ^^ S[  ]S ( MINOS )  icon" icons/piece0" X" " big-icon new  ^^bind Stuhl3
          #1 habox new panel
        #3 vabox new panel
      #1 vabox new panel
    ( [dumpend] ) ;
```

```
31  class;
32
33  : main
34    Spielfeld open-app
35    event-loop bye ;
36  script? [IF]  main  [THEN]
37  previous previous previous
```

Programm 2 — ka2.m

```
1   #! /usr/local/bin/xbigforth
2   \ automatic generated code
3   \ do not edit
4
5   also editor also minos also forth
6
7   component class Spielfeld
8   public:
9     big-icon ptr Stuhl1
10    big-icon ptr Stuhl2
11    big-icon ptr Stuhl3
12    big-icon ptr Stuhl4
13    big-icon ptr Stuhl5
14    big-icon ptr Stuhl12
15    big-icon ptr Stuhl6
16    big-icon ptr Stuhl11
17    big-icon ptr Stuhl10
18    big-icon ptr Stuhl9
19    big-icon ptr Stuhl8
20    big-icon ptr Stuhl7
21
22   ( [varstart] )  ( [varend] )
23  how:
24    : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
25  class;
26
27  Spielfeld implements
28   ( [methodstart] )  ( [methodend] )
29    : widget  ( [dumpstart] )
30              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl1
31              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl2
32              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl3
33              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl4
34              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl5
35            #5 habox new panel
36              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl12
37              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl6
38            #2 habox new panel
39              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl11
40              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl10
41              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl9
42              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl8
43              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl7
44            #5 habox new panel
45          #3 vabox new panel
46        #1 vabox new panel
47      ( [dumpend] ) ;
48  class;
49
50  : main
51    Spielfeld open-app
52    event-loop bye ;
53  script? [IF]  main  [THEN]
54  previous previous previous
```

Programm 3 — ka3.m

```
#! /usr/local/bin/xbigforth
\ automatic generated code
\ do not edit

also editor also minos also forth

component class Spielfeld
public:
  big-icon ptr Stuhl1
  big-icon ptr Stuhl2
  big-icon ptr Stuhl3
  big-icon ptr Stuhl4
  big-icon ptr Stuhl5
  big-icon ptr Stuhl12
  big-icon ptr Stuhl6
  big-icon ptr Stuhl11
  big-icon ptr Stuhl10
  big-icon ptr Stuhl9
  big-icon ptr Stuhl8
  big-icon ptr Stuhl7
 ( [varstart] )  ( [varend] )
how:
  : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
class;

Spielfeld implements
 ( [methodstart] )  ( [methodend] )
  : widget  ( [dumpstart] )
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl1
            $0 $1 *hfil $10 $1 *vfil glue new
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl2
            $0 $1 *hfil $10 $1 *vfil glue new
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl3
            $0 $1 *hfil $10 $1 *vfil glue new
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl4
            $0 $1 *hfil $10 $1 *vfil glue new
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl5
          #9 habox new panel
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl12
            $10 $1 *hfilll $10 $1 *vfil glue new
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl6
          #3 habox new panel
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl11
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl10
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl9
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl8
            ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl7
          #5 habox new panel
        #3 vabox new panel
      #1 vabox new panel
    ( [dumpend] ) ;
class;

: main
  Spielfeld open-app
  event-loop bye ;
script? [IF]  main  [THEN]
previous previous previous
```

Programm 4 — ka4.m

```
#! /usr/local/bin/xbigforth
\ automatic generated code
\ do not edit

```

```
5   also editor also minos also forth
6
7   component class Spielfeld
8   public:
9     big-icon ptr Stuhl1
10    big-icon ptr Stuhl2
11    big-icon ptr Stuhl3
12    big-icon ptr Stuhl4
13    big-icon ptr Stuhl5
14    big-icon ptr Stuhl12
15    big-icon ptr Stuhl6
16    big-icon ptr Stuhl11
17    big-icon ptr Stuhl10
18    big-icon ptr Stuhl9
19    big-icon ptr Stuhl8
20    big-icon ptr Stuhl7
21   ( [varstart] )  ( [varend] )
22  how:
23    : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
24  class;
25
26  Spielfeld implements
27   ( [methodstart] )  ( [methodend] )
28    : widget  ( [dumpstart] )
29              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl1
30              $0 $1 *hfil $10 $1 *vfil glue new
31              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl2
32              $0 $1 *hfil $10 $1 *vfil glue new
33              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl3
34              $0 $1 *hfil $10 $1 *vfil glue new
35              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl4
36              $0 $1 *hfil $10 $1 *vfil glue new
37              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl5
38            #9 habox new panel
39              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl12
40              $10 $1 *hfilll $10 $1 *vfil glue new
41              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl6
42            #3 habox new panel
43              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl11
44              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl10
45              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl9
46              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl8
47              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl7
48            #5 habox new panel
49          #3 vabox new panel
50            ^^ S[  ]S ( MINOS ) X" 4" button new
51            ^^ S[  ]S ( MINOS ) X" 5" button new
52            ^^ S[  ]S ( MINOS ) X" 6" button new
53            ^^ S[  ]S ( MINOS ) X" 7" button new
54            ^^ S[  ]S ( MINOS ) X" 8" button new
55            ^^ S[  ]S ( MINOS ) X" 9" button new
56            ^^ S[  ]S ( MINOS ) X" 10" button new
57            ^^ S[  ]S ( MINOS ) X" 11" button new
58            ^^ S[  ]S ( MINOS ) X" 12" button new
59          #9 habox new panel
60        #2 vabox new panel
61      ( [dumpend] ) ;
62  class;
63
64  : main
65    Spielfeld open-app
66    event-loop bye ;
67  script? [IF]  main  [THEN]
68  previous previous previous
```

Programm 5 — ka5.m

```
#! /usr/local/bin/xbigforth
\ automatic generated code
\ do not edit

also editor also minos also forth

component class Spielfeld
public:
  big-icon ptr Stuhl1
  big-icon ptr Stuhl2
  big-icon ptr Stuhl3
  big-icon ptr Stuhl4
  big-icon ptr Stuhl5
  big-icon ptr Stuhl12
  big-icon ptr Stuhl6
  big-icon ptr Stuhl11
  big-icon ptr Stuhl10
  big-icon ptr Stuhl9
  big-icon ptr Stuhl8
  big-icon ptr Stuhl7
 ( [varstart] ) Variable leds#
 ( [varend] )
how:
  : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
class;

include Spielfeld.fs

Spielfeld implements
( [methodstart] ) : Led ( n n -- )
  case
  0 of Spielfeld Stuhl1 assign endof
  1 of Spielfeld Stuhl2 assign endof
  2 of Spielfeld Stuhl3 assign endof
  3 of Spielfeld Stuhl4 assign endof
  4 of Spielfeld Stuhl5 assign endof
  5 of Spielfeld Stuhl6 assign endof
  6 of Spielfeld Stuhl7 assign endof
  7 of Spielfeld Stuhl8 assign endof
  8 of Spielfeld Stuhl9 assign endof
  9 of Spielfeld Stuhl10 assign endof
 10 of Spielfeld Stuhl11 assign endof
 11 of Spielfeld Stuhl12 assign endof
  endcase ;

: schalten ( n n -- )
\ Nummer Zustand ( 0 = unsichtbar 1 = aus 2 = ein )
  case
  0 of weg swap led endof
  1 of aus swap led endof
  2 of ein swap led endof
  endcase  ;

: leds ( n -- )
  4 - cell * Spielfelder + @
  Max# 0 DO 2 /mod I rot 2dup schalten swap Zustand + c! LOOP
  drop ;

: anzeigen ( -- )
  max# 0 DO I I Zustand + c@ schalten LOOP ;

: Spielzug ( n -- ) umschalten anzeigen ;

 ( [methodend] )
```

```
65
66    : widget  ( [dumpstart] )
67              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl1
68              $0 $1 *hfil $10 $1 *vfil glue new
69              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl2
70              $0 $1 *hfil $10 $1 *vfil glue new
71              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl3
72              $0 $1 *hfil $10 $1 *vfil glue new
73              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl4
74              $0 $1 *hfil $10 $1 *vfil glue new
75              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl5
76            #9 habox new panel
77              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl12
78              $10 $1 *hfilll $10 $1 *vfil glue new
79              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl6
80            #3 habox new panel
81              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl11
82              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl10
83              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl9
84              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl8
85              ^^ S[  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl7
86            #5 habox new panel
87          #3 vabox new panel
88            ^^ S[ 4 leds ]S ( MINOS ) X" 4" button new
89            ^^ S[ 5 leds ]S ( MINOS ) X" 5" button new
90            ^^ S[ 6 leds ]S ( MINOS ) X" 6" button new
91            ^^ S[ 7 leds ]S ( MINOS ) X" 7" button new
92            ^^ S[ 8 leds ]S ( MINOS ) X" 8" button new
93            ^^ S[ 9 leds ]S ( MINOS ) X" 9" button new
94            ^^ S[ 10 leds ]S ( MINOS ) X" 10" button new
95            ^^ S[ 11 leds ]S ( MINOS ) X" 11" button new
96            ^^ S[ 12 leds ]S ( MINOS ) X" 12" button new
97          #9 habox new panel
98        #2 vabox new panel
99      ( [dumpend] ) ;
100   class;
101
102   : main
103     Spielfeld open-app
104     event-loop bye ;
105   script? [IF]  main  [THEN]
106   previous previous previous
```

Programm 6 — ka6.m

```
1   #! /usr/local/bin/xbigforth
2   \ automatic generated code
3   \ do not edit
4
5   also editor also minos also forth
6
7   component class Spielfeld
8   public:
9     big-icon ptr  Stuhl1
10    big-icon ptr  Stuhl2
11    big-icon ptr  Stuhl3
12    big-icon ptr  Stuhl4
13    big-icon ptr  Stuhl5
14    big-icon ptr  Stuhl12
15    big-icon ptr  Stuhl6
16    big-icon ptr  Stuhl11
17    big-icon ptr  Stuhl10
18    big-icon ptr  Stuhl9
19    big-icon ptr  Stuhl8
20    big-icon ptr  Stuhl7
21   ( [varstart] ) Variable leds#
```

```
22   ( [varend] )
23  how:
24    : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
25  class;
26
27  include Spielfeld.fs
28  Spielfeld implements
29   ( [methodstart] ) : Led ( n n -- )
30    case
31    0 of Spielfeld  Stuhl1 assign endof
32    1 of Spielfeld  Stuhl2 assign endof
33    2 of Spielfeld  Stuhl3 assign endof
34    3 of Spielfeld  Stuhl4 assign endof
35    4 of Spielfeld  Stuhl5 assign endof
36    5 of Spielfeld  Stuhl6 assign endof
37    6 of Spielfeld  Stuhl7 assign endof
38    7 of Spielfeld  Stuhl8 assign endof
39    8 of Spielfeld  Stuhl9 assign endof
40    9 of Spielfeld  Stuhl10 assign endof
41   10 of Spielfeld  Stuhl11 assign endof
42   11 of Spielfeld  Stuhl12 assign endof
43    endcase ;
44
45  : schalten ( n n -- )
46  \ Nummer Zustand ( 0 = unsichtbar 1 = aus 2 = ein )
47    case
48    0 of weg swap led endof
49    1 of aus swap led endof
50    2 of ein swap led endof
51    endcase  ;
52
53  : leds ( n -- )
54    4 - cell * Spielfelder + @
55    Max# 0 DO 2 /mod I rot 2dup schalten swap Zustand + c! LOOP
56    drop ;
57
58  : anzeigen ( -- )
59    max# 0 DO I I Zustand + c@ schalten LOOP ;
60
61  : Spielzug ( n -- ) umschalten anzeigen ;
62
63   ( [methodend] )
64    : widget  ( [dumpstart] )
65              ^^ S[ 0 Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl1
66              $0 $1 *hfil $10 $1 *vfil glue new
67              ^^ S[ 1  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl2
68              $0 $1 *hfil $10 $1 *vfil glue new
69              ^^ S[ 2 Spielzug  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl3
70              $0 $1 *hfil $10 $1 *vfil glue new
71              ^^ S[ 3  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl4
72              $0 $1 *hfil $10 $1 *vfil glue new
73              ^^ S[ 4 Spielzug  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl5
74            #9 habox new panel
75              ^^ S[ 11  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl12
76              $10 $1 *hfilll $10 $1 *vfil glue new
77              ^^ S[ 5  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl6
78            #3 habox new panel
79              ^^ S[ 10  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl11
80              ^^ S[  9  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl10
81              ^^ S[  8  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl9
82              ^^ S[  7  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl8
83              ^^ S[  6  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind  Stuhl7
84            #5 habox new panel
85          #3 vabox new panel
86            ^^ S[ 4 leds ]S ( MINOS ) X" 4" button new
87            ^^ S[ 5 leds  ]S ( MINOS ) X" 5" button new
```

```
88            ^^ S[ 6 leds  ]S ( MINOS ) X" 6" button new
89            ^^ S[ 7 leds  ]S ( MINOS ) X" 7" button new
90            ^^ S[ 8 leds  ]S ( MINOS ) X" 8" button new
91            ^^ S[ 9 leds  ]S ( MINOS ) X" 9" button new
92            ^^ S[ 10 leds  ]S ( MINOS ) X" 10" button new
93            ^^ S[ 11 leds  ]S ( MINOS ) X" 11" button new
94            ^^ S[ 12 leds  ]S ( MINOS ) X" 12" button new
95          #9 habox new panel
96        #2 vabox new panel
97      ( [dumpend] ) ;
98  class;
99
100 : main
101   Spielfeld open-app
102   event-loop bye ;
103 script? [IF]  main  [THEN]
104 previous previous previous
```

Programm 7 — ka7.m

```
1   #! /usr/local/bin/xbigforth
2   \ automatic generated code
3   \ do not edit
4
5   also editor also minos also forth
6
7   component class Spielfeld
8   public:
9     big-icon ptr Stuhl1
10    big-icon ptr Stuhl2
11    big-icon ptr Stuhl3
12    big-icon ptr Stuhl4
13    big-icon ptr Stuhl5
14    big-icon ptr Stuhl12
15    big-icon ptr Stuhl6
16    big-icon ptr Stuhl11
17    big-icon ptr Stuhl10
18    big-icon ptr Stuhl9
19    big-icon ptr Stuhl8
20    big-icon ptr Stuhl7
21   ( [varstart] ) Variable leds#
22   ( [varend] )
23  how:
24    : params   DF[ 0 ]DF X" König Artus Tafelrunde" ;
25  class;
26
27  include Spielfeld.fs
28  Spielfeld implements
29   ( [methodstart] ) : Led ( n n -- )
30    case
31    0 of Spielfeld  Stuhl1 assign endof
32    1 of Spielfeld  Stuhl2 assign endof
33    2 of Spielfeld  Stuhl3 assign endof
34    3 of Spielfeld  Stuhl4 assign endof
35    4 of Spielfeld  Stuhl5 assign endof
36    5 of Spielfeld  Stuhl6 assign endof
37    6 of Spielfeld  Stuhl7 assign endof
38    7 of Spielfeld  Stuhl8 assign endof
39    8 of Spielfeld  Stuhl9 assign endof
40    9 of Spielfeld  Stuhl10 assign endof
41   10 of Spielfeld  Stuhl11 assign endof
42   11 of Spielfeld  Stuhl12 assign endof
43    endcase ;
44
45  : schalten ( n n -- )
46  \ Nummer Zustand ( 0 = unsichtbar 1 = aus 2 = ein )
```

```
47    case
48    0 of weg swap led endof
49    1 of aus swap led endof
50    2 of ein swap led endof
51    endcase  ;
52
53  : leds ( n -- )
54    4 - cell * Spielfelder + @
55    Max# 0 DO 2 /mod I rot 2dup schalten swap Zustand + c! LOOP
56    drop ;
57
58  : anzeigen ( -- )
59    max# 0 DO I I Zustand + c@ schalten LOOP ;
60
61  : Spielzug ( n -- ) umschalten anzeigen ;
62
63   ( [methodend] )
64    : widget  ( [dumpstart] )
65              ^^ S[ 0 Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl1
66              $0 $1 *hfil $10 $1 *vfil glue new
67              ^^ S[ 1  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl2
68              $0 $1 *hfil $10 $1 *vfil glue new
69              ^^ S[ 2 Spielzug  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl3
70              $0 $1 *hfil $10 $1 *vfil glue new
71              ^^ S[ 3  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl4
72              $0 $1 *hfil $10 $1 *vfil glue new
73              ^^ S[ 4 Spielzug  ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl5
74            #9 habox new panel
75              ^^ S[ 11  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl12
76              $10 $1 *hfilll $10 $1 *vfil glue new
77              ^^ S[ 5  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl6
78            #3 habox new panel
79              ^^ S[ 10  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl11
80              ^^ S[  9  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl10
81              ^^ S[  8  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl9
82              ^^ S[  7  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl8
83              ^^ S[  6  Spielzug ]S ( MINOS )  icon" LED_ge_aus" X" " big-icon new  ^^bind Stuhl7
84            #5 habox new panel
85          #3 vabox new panel
86            ^^ #12 #8 SC[ ( pos -- ) leds ]SC ( MINOS ) hscaler new  #4 SC#
87          #1 habox new
88        #2 vabox new panel
89      ( [dumpend] ) ;
90  class;
91
92  : main
93    Spielfeld open-app
94    event-loop bye ;
95  script? [IF]  main  [THEN]
96  previous previous previous
```

Ein ANS–Plädoyer für FORGET

Fred Behringer

In Turbo–Forth (16 Bit) gibt es für solche Nebenrechnungen im Interpreter–Modus, die Kontroll–Strukturen (Schleifen) beanspruchen, das Wort

```
:: ... Worte wie im Parameterfeld
           einer Colon-Definition ... ;
```

Hier ein Anwendungsbeispiel:

```
decimal ↩  ok
:: 20 20 1 do i + loop . ; ↩  210 ok
```

In diesem Beispiel werden die Zahlen von 1 bis 20 aufaddiert und das Ergebnis, nämlich 210, wird auf dem Bildschirm ausgegeben. Das Beispiel soll nur der Demonstration dienen. Man überlegt sich ja leicht, wie die Ergebniszahl 210 zustande kommt: Man schreibe sich die Zahlenfolge einmal vorwärts und einmal rückwärts untereinander hin. Die übereinanderstehenden Zahlen ergeben jedes Mal 21. Und das 20 mal. Verlangt ist die Summe der Reihe nur ein einziges Mal, gleichgültig, ob vorwärts oder rückwärts aufaddiert. Also muss noch durch 2 geteilt werden. Ergebnis: 210.

In ZF ist `::` nicht vorgesehen. Man kann aber die Definition von `::` aus Turbo–Forth auch für ZF verwenden. Sie lautet:

```
: :: hide here >r [ ' : @ ] literal ,
  !csp ] r@ execute r> dp ! ;
```

Das Ganze geht auf F83 zurück und Turbo–Forth wie auch ZF sind F83–Derivate. MinForth von Andreas Kochenburger (zumindest in der Version 1.5) weigert sich, diese Definition von `::` zu akzeptieren. Wie soll es auch? Von `dp` ist in ganz ANS–Forth keine Rede, von `!csp` ebenso wenig. `hide` ist eine schöne Sache, aber in den ANS–Empfehlungen gibt es kein `hide`.

Nun gibt es aber doch das interessante Wort `forget` — in Turbo–Forth und in ZF, aber auch in MinForth. Und wie man hört, könnte `forget` im nächsten ANS–Standard wieder aufgenommen werden. Selbstverständlich könnte ich das gesamte Parameterfeld voller cfa–Werte in eine voll ausgebildete Colon–Definition mit Namen <name> packen und nach Aufruf von <name> ↩ und Abarbeiten der im Parameterfeld von <name> liegenden cfa–Werte dann `forget` <name> anwenden. Das würde zwar von Seiten des Anwenders etwas mehr Disziplin verlangen als das mechanisch anwendbare `::` aus Turbo–Forth, wäre aber weniger gekünstelt: Man hat ja plastisch vor Augen, was geschieht. Oder anders ausgedrückt: Man müsste sich überlegen, wie das, was man vorhat, zustande kommt, um die vorgeschlagene Konstruktion immer richtig hinschreiben zu können. Und es erscheinen, von `forget` oder `marker` mal abgesehen, nur Worte aus dem ANS–Core–Word–Set, also auch für jeden Nicht–System–Implementierer mehr als einleuchtend:

```
: <name> ... ; <name> forget <name> ↩  ok
```

Das funktioniert in jedem System, das `forget` in einer Form enthält, die seinem Namen gerecht wird. <name> kann beliebig gewählt werden. MinForth ist 32 Bit breit, in C geschrieben und unterprogramm–gefädelt. Spielt alles keine Rolle! Ich schreibe das obige Beispiel noch einmal hin, diesmal mit der vorgeschlagenen Syntax:

```
decimal ↩  ok
: :: 20 20 1 do i + loop . ; :: forget :: ↩
210 ok
```

Natürlich kann man das auch mit Systemen machen, die den noch aktuellen ANS–Empfehlungen entsprechen, z. B. Gforth 0.6.2, wenn man statt zu `forget` seine Zuflucht zu `marker` nimmt:

```
decimal ↩  ok
marker ;; ↩  ok
: :: 20 20 1 do i + loop . ; :: ;; ↩  210 ok
```

Vielleicht lässt sich die folgende Version leichter merken, da sie mit `: :: ... ;; ;` einen etwas symmetrischeren Eindruck hinterlässt:

```
decimal ↩  ok
marker ;; ↩  ok
: :: 20 20 1 do i + loop . ;; ; :: ↩  210 ok
```

Und auch bei dem Beispiel mit `forget` kann man das eliminierende Wort, in diesem Fall `forget`, in die Colon–Definition mit hineinziehen:

```
decimal ↩  ok
: :: 20 20 1 do i + loop . forget ; :: :: ↩
210 ok
```

Leichter wird es dadurch allerdings kaum: Das erste `::` nach dem `;` ruft hier die Colon–Definition `::` auf, der *Inhalt* der Colon–Definition wird abgearbeitet und dabei tritt als letztes Wort `forget` in Aktion, das das zweite, also das noch im Eingabestrom verbliebene `::` hinter `;` abarbeitet — womit alles wieder so ist wie bei Eintritt in die Colon–Definition über `:` .

Nachwort: In Forth kann man auch als Anwender (mit den Mitteln der Sprache) ins System eingreifen. Im ANS–94–Dokument heißt es hierzu: One of the features of Forth that has endeared it to its users is that the same tools that are used to implement the system are available to the application programmer — a result of this approach is the compactness and efficiency that characterizes most Forth implementations.

Nun, mir war dieses Feature keine große Hilfe. Die für das Wort `::` aus Turbo–Forth eingesetzten Tools konnten allein mit den Mitteln der Sprache gar nicht ganz allgemein nach ANS–Forth übertragen werden. Sie unterscheiden sich, wenn man Forth als konkret anwendbares System und nicht abstrakt als Sprache betrachtet,

zu stark von System zu System. *Back to the roots* lautet das Zauberwort. Nicht (mit unzugänglichen Forth–Worten) am System herumfeilen, sondern an den Überlegungen (zu dem, was eigentlich gemacht werden soll). Dem Standardisierungs–Bestreben tut das keinen Abbruch: Wenn ich an System–Worten wie `::` herumdoktere, bin ich, obwohl ich das aus einem Anwender–Programm heraus machen kann, bestimmt kein reiner *Application–Programmer* mehr.

Jiffy

Michael Kalus

Im Heft 1/2010 wurde am Beispiel des amforth für den atmega168 beschrieben, wie einem Forth für Mikrocontroller ein Schrittmacher eingesetzt werden kann. In der Timer–Interrupt–Service–Routine wurde ein Phasenakkumulator benutzt, um eine im Mittel genaue Sekunde zu generieren. Wie im Listing zu sehen war, ging diese Uhr zu schnell. Der geschätzte Wert für den Phasenakku — der JIFFY — war zu ungenau. Inzwischen benutze ich einen 3 Byte breiten Phasenakku, berechne den Jiffy, und bestimme eine Korrektur für den Wert — siehe Listing (gforth). Zudem ist Jiffy im amforth nun als Variable angelegt, damit die Ganggenauigkeit der Uhr vom Terminal aus nachgestellt werden kann[1]. Die Berechnung erfolgt als Gleitkommazahl, um ein Gefühl dafür zu vermitteln, dass die Ganggenauigkeit mit diesem Verfahren ja nicht beliebig hoch werden kann, weil immer ein Rest bleibt. Für meinen Hausgebrauch finde ich es aber ok, wenn die Uhr in den Steuerungen eine Sekunde pro Tag ungenau geht.

Links

`http://en.wikipedia.org/wiki/Jiffy_(time)`
`http://de.wikipedia.org/wiki/Arithmetik`
`http://de.wikipedia.org/wiki/Gleitkommazahl`

Listing

```
1   vocabulary jiffywords jiffywords definitions
2   decimal
3   : .. bye ;
4
5   : d*     ( ud1 ud2 -- udprod )
6     >r swap >r 2dup um*
7     2swap r> * swap r> * + + ;
8
9   \ jiffy berechnen.
10  &001 s>d    \ phasenaccu Breite:
11  &256 s>d d* \ byte0
12  &256 s>d d* \ byte1
13  &256 s>d d* \ byte2
14  &256 s>d d* \ prescaler
15  &256 s>d d* \ Takte bis timer2 overflow
16  2constant tickers
17  2variable mhz  20.000000 mhz 2!
18  2variable cnt        0 0 cnt 2!
19
20  : cnt+      cnt 2@  1 s>d d+ cnt 2! ;
21  : cnt-      cnt 2@  1 s>d d- cnt 2! ;
22  : +mhz      ( tickers -- tickers+mhz ) mhz 2@ d+ ;
```

[1] Die Datei tim2ISR.asm gebe ich auf Anfrage gerne weiter.
michael.kalus@onlinehome.de

```
23  : -mhz      ( tickers -- tickers-mhz ) mhz 2@ d- ;
24  : jiffy     ( -- rest jiffy )
25               0 0 cnt 2! tickers
26              begin -mhz cnt+ 2dup d0< until +mhz cnt- cnt 2@  ;
27
28  \ Interval zwischen zwei Zeiten in Sekunden:
29  : mm    60 * ;
30  : hh    60 * mm ;
31  : dd    24 * hh ;
32  : sec { d h m s -- s }  \ convert day hour minute second into seconds
33        0 d dd +   h hh +   m mm +   s + ;
34  : interval (  d h m s   d h m s - v )  sec >r sec r> swap - ;
35
36  \ Verhältnis der Sekundenabweichung zum Zeitinterval in ppm:
37  : ppm   ( v s -- ppm ) 1000000 * swap / ;
38
39  \ Korrektur des jiffy:
40  \ v = interval in Sekunden, s = Gangabweichung in Sekunden, j0 = altes jiffy.
41  : j     { v s j0 -- j1 }
42          s s>d d>f  v s>d d>f f/
43          1e0 f+ j0 s>d d>f f*  ;
44  : j.    j f. ;
45
46  cr cr
47
48  .( Jiffy = )
49   3 16 56 40  4 8 54 11 interval
50   1 -54968 j.
51
52  cr cr
53  \ finis
```

Forth–Gruppen regional

Mannheim
Thomas Prinz
Tel.: (0 62 71) – 28 30 (p)
Ewald Rieger
Tel.: (0 62 39) – 92 01 85 (p)
Treffen: jeden 1. Dienstag im Monat
Vereinslokal Segelverein Mannheim e.V. Flugplatz Mannheim–Neuostheim

München
Bernd Paysan
Tel.: (0 89) – 46 22 14 91 (p)
`bernd.paysan@gmx.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00, im Sommer (Mai–September) im Chilli Asia Dachauer Str. 151, im Winter im Sloveija Grill, Dachauer Str. 147, 80335 München.

Hamburg
Küstenforth
Klaus Schleisiek
Tel.: (0 40) – 37 50 08 03 (g)
`kschleisiek@send.de`
Treffen 1 Mal im Quartal
Ort und Zeit nach Vereinbarung (bitte erfragen)

Mainz
Rolf Lauer möchte im Raum Frankfurt, Mainz, Bad Kreuznach eine lokale Gruppe einrichten.
Mail an rowila@t-online.de

Gruppengründungen, Kontakte

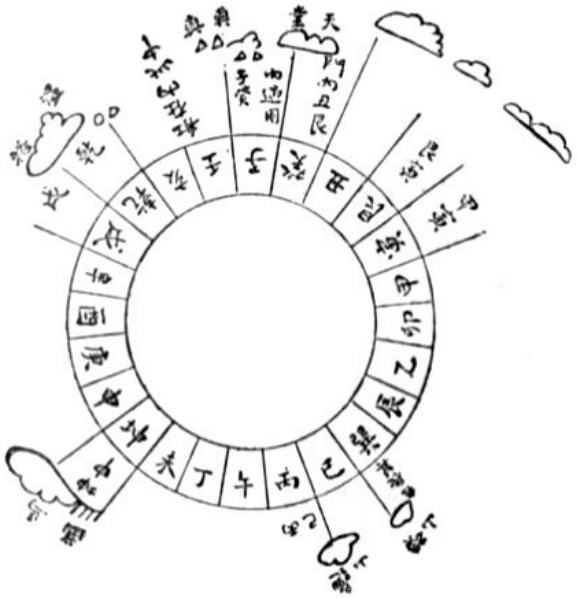
Hier könnte Ihre Adresse oder Ihre Rufnummer stehen — wenn Sie eine Forthgruppe gründen wollen.

µP–Controller Verleih

Carsten Strotmann
`microcontrollerverleih@forth-ev.de`
`mcv@forth-ev.de`

Spezielle Fachgebiete

FORTHchips (FRP 1600, RTX, Novix)	**Klaus Schleisiek–Kern** Tel.: **(0 40) – 37 50 08 03** (g)
KI, Object Oriented Forth, Sicherheitskritische Systeme	**Ulrich Hoffmann** Tel.: **(0 43 51) – 71 22 17** (p) Fax: **– 71 22 16**
Forth–Vertrieb volksFORTH ultraFORTH RTX / FG / Super8 KK–FORTH	Ingenieurbüro **Klaus Kohl–Schöpe** Tel.: **(0 70 44)–90 87 89** (p)



Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen? Schreiben Sie einfach der VD — oder rufen Sie an — oder schicken Sie uns eine E–Mail!

Hinweise zu den Angaben nach den Telefonnummern:
Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich
Die Adressen des Büros der Forth–Gesellschaft e.V. und der VD finden Sie im Impressum des Heftes.

Einladung zur

Forth–Tagung 2011

vom 15. bis 17. April 2011

im Bildungshaus Zeppelin

Zeppelinstraße 7, 38640 Goslar

http://www.bildungshaus-zeppelin.de

Geplantes Programm

Donnerstag, 14.04.2011
nachmittags Treffen der Frühankommer
Forth–200x–Standard–Treffen

Freitag, 15.04.2011
nachmittags Beginn der Forth–Tagung
Vorträge und Workshops

Samstag, 16.04.2011
vormittags Vorträge und Workshops
nachmittags Exkursion

Sonntag, 17.04.2011
09:00 Uhr Mitgliederversammlung
nachmittags Ende der Tagung

Anreise siehe: http://www.bildungshaus-zeppelin.de/Anreise.156.0.html

Die Innenstadt von Goslar

Quellen: www.bildungshaus-zeppelin.de, wikimedia.org