Vierte Dimension

2/2017

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

Titelbild Die in *Clock Works 2* (S. 12ff) vorgestellte Anzeige der Zeit in einer dem Abakus nachempfundenen Darstellung. Das im Bild obere Paar LED–Reihen zeigt die Stunden. Es ist 18:18:47 Uhr. *E. Wälde*

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,

ist es Euch schon aufgefallen, dass wir, *Gerald Wodni, Bernd Paysan und ich*, die neue Webpräsenz des Vereins vorantreiben? Nein? Das könnte daran liegen, dass man vom *neuen Blog* auch noch nichts Offizielles sehen kann. Doch im *Wiki* hat sich im Mai des Jahres eine ganze Menge getan! Der Umzug der Sammlungen und statischen Seiten von der alten Website dorthin ist schon weit gekommen und kann bereits durchstöbert werden. Also immer mal wieder reinzappen ins Forth–Wiki. Der Blog, das Forth–Wiki und theforth.net als Quelle von Programmcode sollen ja mal die drei tragenden Säulen des Forth sein.

Wo wir grad beim Aufräumen sind, schau mal in Deinen Kleiderschrank! Forth–T–Shirt gefunden? Meins müsste auch mal durch ein neues ersetzt werden, was ich da gefunden habe, ist doch schon arg verwaschen! Also auf gehts, *Bernd* macht grad neue Forth–T–Shirt und ihr könnt anfangen zu bestellen!

Testumgebungen bauen, immer wieder neu. Auch darin kann man Routine bekommen. Wie man dann aber an seinem 1–Draht das ganze Hühnerfutter an Sensoren wieder findet, das man da neulich drangelötet hat, zeigt uns *Rafael Deliano*. Damit wird es dann doch nicht die mühsame Suche nach der berühmten Stecknadel im Heuhaufen, sondern eine glatte Sache.

Seine besondere Vorliebe für ausgefallene Uhren hat ihm schon manches Aha–Erlebnis beschert. *Erich Wälde* lässt euch wieder teilhaben, auch an der Tücke des Objekts. Und weil er immer weiß, was die Stunde geschlagen hat, lieferte er auch gleich das Protokoll der Mitgliederversammlung und hat alles fein säuberlich ins Heft eingebaut. Da brauchte ich gar nichts mehr zu tun! Und Bernd machte es gleich auch so auch. Ihr Lieben, mit eurer Hilfe war das Heft diesmal sehr schnell fertig! Vielen Dank dafür.

Komfortables Forth haben, und dennoch kleine Targets in der MCU erzielen? *Klaus Schleisiek* hat sich auf den Weg gemacht, das für seine μCore–Targets zu lösen. Auf der Tagung in Kalkar entfachte das Thema eine rege Diskussion um die verschiedenen Ansätze „wie man das löst“. Hier sein Standpunkt. Bin gespannt, was da noch nachkommt. Ting geht da den radikalen Weg, alles ganz simpel zu halten. Er möchte simpler werden, nicht komplexer.

Kommen wir zum #tag. Im Deutschen sagen wir dazu einen Gegenstand auszeichnen, beschildern oder markieren. Beim Obst und Gemüse und Waren an sich macht es Sinn, ein Preisschild dran zu haben. Denken wir hierzulande jedenfalls. In anderen Ländern ist das anders, da muss man über den Preis reden. Aber soll man Forth–Worte mit tags versehen? Soll das in Rot geschriebene Wort etwas anderes machen als das in Grün geschriebene? *Matthias Trute* hat sich damit näher befasst.

Im September ist die 33. EuroForth, diesmal wieder in Österreich. Eine Institution schon so alt wie die Forthgesellschaft selbst. Nehmt extra Zeit mit, wenn ihr dahin fahrt, denn Östereich und Wien sind immer eine Reise wert!

Bis dahin!
Euer Michael

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können sie auch von der Web–Seite des Vereins herunterladen.
`http://fossil.forth-ev.de/vd-2017-02`

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

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

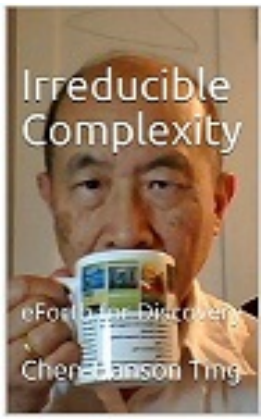
Ein bemerkenswerter Zufall

Aus zwei Nationen waren Menschen zur Forthtagung im April am letzten Osterferien–Wochenende nach Kalkar angereist. Zwei davon waren eingeladen worden, um etwas über ihre Vergangenheit mit und in Kalkar vorzutragen — sie hatten damals dort gearbeitet. Wie sich später dann noch herausstellte, war ein weiterer Tagungsteilnehmer Augenzeuge, und vor vielen Jahren in Kalkar gewesen, aber um gegen das AKW zu demonstrieren. Und der Vater eines anderen Tagungs-Teilnehmers war Polizist und schützte das AKW vor den Demonstranten. Und noch ein Augenzeuge war unter uns: Er war in den ersten Tagen des Abbruchs in Kalkar, um dort als bezahlter Fotograf Bilder für ein TV–Spiel zu machen. Also fünf von 21 (bzw 25) Menschen waren vor Jahren schon mal in Kalkar gewesen!

Gruß euch allen! Martin

Dire Consequence of Moore's Law

Von Chen–Hanson Ting erschienen ist im Mai 2017 ein weiteres Ebook: *Irreducible Complexity — eForth for Discovery*. Herausgegeben von Jürgen Pintaske[1]. Ting befasst sich darin mit den Vorzügen von eForth, ganz besonders, wenn es um MCUs geht, und was er dabei schon so alles erlebt hat!



Im Abschnitt 1.3 schreibt Ting:

> A while ago, I was amazed at the 566 page reference manual of ATmega328 from Atmel, which is a lowly 8-bit microcontroller used on Arduino Uno Kit. The reference manual of STM32F407 is 1713 pages thick. How can anybody wading through this document to get a handle on this chip and all its peripheral devices?
>
> I opened the Demo project for the STM32F407–Discovery Board on Keil's uVision5. In the Project panel I counted 7 folders with 31 files in them. Just for a Demo! It is true that the Demo does a lot of interesting things, like reading the 3–axis accelerometer and the USB connection to PC. I have great sympathy for people who gets this kit and is confronted by this huge software mess.
>
> The dire consequence of the Moore's Law is complexity beyond comprehension.[2]
>
> The only way to deal with this complexity is the Forth way. Or, put it more bluntly: KISS — Keep It Simple, Stupid!
>
> The first thing to do is to put eForth on board. The 16 MHz high speed internal clock HSI in the chip is good enough for an USART. Forget about the fancy PLL that can push the clock to 168 MHz. We can deal with it when we really need the speed. Just get the USART going, and we can walk into the guts of the microcontroller and actually control it from inside through eForth.
>
> What about interrupts, threads, heaps, multitasking, and preemptive task switching? All the great things this ARM/THUMB chip can do? Forget them! You will learn them in the senior year of computer engineering, if you have time to go to school. All these things can be added to eForth when you really need them. eForth exposes all the memory and the IO registers to you. You can inspect them, and you can tinker with them.
>
> This is the way to study the peripheral devices, learn how to control them, and make use of them. Focus on one device you will use. Read that chapter in the reference manual. Inspect the status and control registers. Flip bits in the control register and see what happens. Write short commands to perform the functions you want. These functions will be called from you eventual applications ...[3]

Viel Vernügen mit der Lektüre und praktischen Erfolg damit danach. mk

amforth 6.5

> A new release of amforth is now available. It fixes a really nasty bug in the interrupt handling on the AVR plattform. Erich did an outstandig job discovering and fixing the bug, that sometimes made interrupts simply disappear. This happened very seldom and under very limited cirumstances (active multitasker heavy timer interrupts). I took the opportunity to apply his idea to the MSP430 plattform. For now the G2553 (found in the popular launchpads) has a special hex file that has support for interrupts just as the AVR's have for a long time: High level Forth words. It's in it's very early stages so be careful when using it.
>
> Have fun, Matthias[4]

[1] Auf dem Cover sieht man Ting mit Jürgens Forth–Mug :-)

[2] „Die schlimme Folge des Moore'schen Gesetzes ist die Komplexität jenseits des Verständnisses."

[3] https://wiki.forth-ev.de/doku.php/projects:ting_s_electronic_forth_bookshelf

[4] https://sourceforge.net/p/amforth/mailman/message/35814877/

Nur 4 Wochen nach 6.4 schon ein neuer Release? Das hat durchaus was mit der „Anzeige a la Abakus“ hier im Heft zu tun. Und mit der „Verlorenen Zeit“, doch die findet ihr erst im nächsten Heft... Erich

Das Alles in einer Tasche

Radio, Walkman, Videorecorder, Spielkonsole, Taschenrechner, Telefon, Uhr, Kompass, Wasserwaage, Funksprechanlage — das alles passt heute locker *zusammen* in eine Hemdtasche. In meiner Rumpelkammer gibt es tatsächlich auch noch einiges davon. Einige Langspielplatten, und eine Schachtel mit Tonbandkasetten, und sogar Disketten. Und einen C64. Kinder wie die Zeit vergeht.

Vielleicht sollte ich auch mal auf die Vintage nach München gehen. Zum achtzehnten Mal war das *Vintage Computer Festival* am Wochenende vom 29. April bis zum 1. Mai 2017 im schönen München. Ziel des VCF ist es, den Erhalt und die Pflege historischer Computer und anderer (E)DV Gerätschaften zu fördern, das Interesse in „überflüssige“ Hard- und Software zu wecken und vor allem den Spaß daran auszuleben.[5] Wer war da? mk

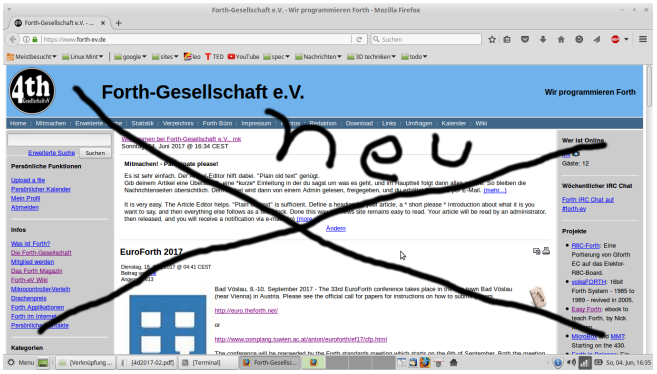
`www.forth-ev.de` aufräumen

Im Mai hat sich da einiges getan hinter den Kulissen unserer alten geeklog–Datenbank, mit der `www.forth-ev.de` betrieben wird. Weniger Technisches diesmal, dafür Inhaltliches. Die ganze gute alte website wurde revidiert: Was kann weg, was muss bleiben und was passt dann in welche Form? Es wurde klar: Künftig wird es nur die beiden getrennten Formate *blog* und *wiki* geben. Und nicht mehr so ein Versuch einer eierlegenden Wollmilchsau aus Nachrichten, Sammlungen, Statistiken, Links, Infoseiten und Reklame auf den Banden. Inzwischen ist der ganze alte Haufen gesichtet worden, und es wurde beschlossen, was wo hin umziehen soll. Und dann ging es damit auch gleich los. Eine Menge an Daten sind schon in ihre neuen Formate kopiert.

Vieles aus der alten website waren „statische Seiten“ — wie die ganze Sammlung unserer „Vierten Dimension“, Listings und so etwas aus dem „Download–Bereich“. Soetwas lässt sich viel besser im wiki sammeln als in so einer geeklog–Datenbank. So sind nun alle Hefte gut aufgehoben in unserem wiki, und auch so statische Seiten wie die „Drachenträger“ sind nun dort versammelt und andere „Infos“ — aber seht selbst: `https://wiki.forth-ev.de/doku.php`.

Die Nachrichten hingegen, denen ja nur ein Zeitwert zukommt, erhalten das neue blog–Format. Etliche Nachrichten sind schon gewandelt. Offen ist noch, ob die starke Unterteilung in Kategorien, die wir da hatten, überhaupt Sinn macht. Ich möchte das nicht mitnehmen, denn die Revision hat auch ergeben, dass davon überhaupt kein Gebrauch gemacht wurde in der Vergangenheit. Einfach nur „Nachrichten“, sonst nichts, das genügt. Und eine „Startseite“, auf der aktuelle Nachrichten sich eine Weile halten. Eine Suchfunktion für alte Nachrichten fehlt dann noch — so eine Art automatisches Stichwortverzeichnis wäre was!

Noch ist offen, wann offiziell umgeschaltet werden kann. Vor allem technische Fragen zur Navigation und zur Rechte-Verwaltung auf dem blog sind noch zu klären. Aber dann kann es losgehen. mk



[5] http://vcfe.org/D/

Neue Drachen–Shirts

Bernd Paysan

Die vor fünf Jahren designten Shirts mit chinesischem Drachen haben lang gehalten, sind aber doch sichtbar ausgebleicht und der Stoff ist auch löchrig. Deshalb müssen neue her.

Abbildung 1: Drachenshirt weiß

Abbildung 2: Drachenshirt schwarz

So sahen sie vor 5 Jahren aus

Die frisch gedruckten T–Shirts habe ich am Vortag vor der Fahrt zum LinuxTag nach Berlin gleich fotografiert, und an das Team verschickt. Das verwendete T–Shirt hat eine Brusttasche, in die der Nerd dann seine Stifte 'reintun kann.

Die T–Shirts waren damals schon die „coolsten T–Shirts" auf dem LinuxTag, behauptet zumindest Erich Wälde, am Belug–Stand erfahren zu haben ;-), und entsprechendes Lob kam auch auf anderen Veranstaltungen.

Auch haben wollen?

Die bisher gedruckten Shirts sind längst weg, deshalb müssen mehr her, Preis pro Shirt mit Druck ca. 20€, ich werde das zum Selbstkostenpreis abwickeln.

Der Textildrucker hat einen umfangreichen Katalog, es kann also alles Mögliche bedruckt werden: `http://www.textileworld.eu/catalogsearch/result/`, vom Handtuch bis zur Laptoptasche ist da so ziemlich alles dabei, auch die Handtuch-Variante sieht gut aus. Wir haben bei der letzten Bestellung aber viel zu viele Varianten bestellt, deshalb gebe ich dieses Mal vor, was bestellt werden darf. Das ist vielleicht auf der Tagung nicht richtig 'rübergekommen, obwohl ich es gesagt habe: Jeder Wechsel des Stoffs macht Mühe. Jeder Wechsel des Motivs verursacht noch erheblich mehr Aufwand. Die Technik ist zwar für Einzelanfertigungen geeignet, trotzdem fällt dabei ein Mehraufwand an. Und dann schieben die unseren Auftrag auf „später, wenn nichts los ist", und ziehen andere Aufträge vor.

Ich werde dieses Mal JN 920 Weiß oder Schwarz bestellen, die beim letzten Mal populären Biosfair–T–Shirts sind nicht mehr verfügbar. Für die farbenfrohen Damen ggf. 16.1224 | Fruit of the Loom Lady–Fit Value V–Neck T.

Seriennummern der ICs im Dallas–Bus finden

Rafael Deliano

In sehr kleinen Messobjekten mit mehreren Sensoren ist wenig Platz. Getrennte Aderpaare zu jedem Sensor sind dann nicht mehr möglich, und man muss auf einen passenden Bus übergehen. Der Dallas–1–wire–Bus[1] bietet sich an. Oft kann man den GND[2] des Messobjekts mitverwenden (Abb. 1) und man hat dann wirklich eine „one wire“–Verbindung. Durch die digitale Datenübertragung ergibt sich keine Beeinträchtigung des Messwerts. Die kleine U–Bauform[3] bietet sich besonders an (Abb.2). Sie hat durch ihre geringe Masse und die vielen NC–Pins[4], die man z.B. auf GND–Flächen der Platine löten kann, auch thermische Vorteile. Hinweis: Grundlagenwissen zum Dallas–Bus findet ihr in [1] bis [3] und die Beschreibung der verwendeten Treiber im Beitrag „Temperatur–Logger“ in [4].

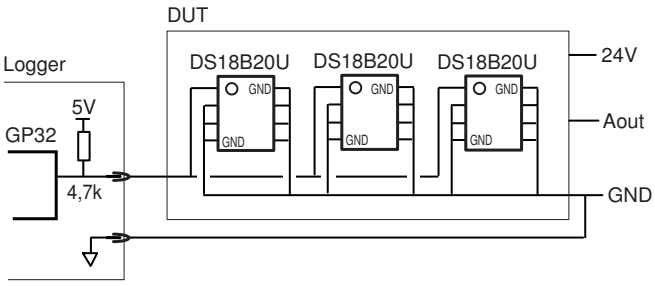


Abbildung 1: Bus–Prinzip

Abbildung 2: U–Bauform mit Farbpunkt; Stecknadelkopf zum Vergleich

Sensoren per Seriennummern ansteuern

Für die Ansteuerung im Bus ist es nötig, die 64–Bit–Seriennummer des Sensors zu kennen. Die ist eigentlich nur 6 Bytes lang, weil Typcode und CRC enthalten sind (Abb. 3), es ist aber angenehmer, mit den vollen 8 Byte zu arbeiten.

Die simple Variante ist es, die ICs erstmal einzeln zu verdrahten und dabei jeweils deren Seriennummer auszulesen und zu speichern. Es ist nützlich, dann die Sensoren individuell zu kennzeichnen. Farbpunkt bietet sich dafür an. Man trägt dazu den Acryllack[5] nicht mit dem Pinsel, sondern mit einer Stecknadelspitze auf. Der Aufwand für dauerhafte Markierungen lohnt sich, denn die Bauform ist arg teuer, ca. 4 EUR. So kann man die Teile besser wiederverwenden, wenn man, wie ich, häufig Testaufbauten von begrenzter Lebensdauer herstellen muss.

Da ich auch Fotos mache, die zeigen, wo genau die Sensoren im Gerät angebracht wurden, hat sich der Farbpunkt als besonders entwicklerfreundlich für die Zuordnung erwiesen.

Es gibt aber die Alternative, unmarkierte Temperatur–Sensoren zu verdrahten, und dann mit dem *Search–ROM–Algorithmus* alle Seriennummern zu lesen. Danach wird von allen zyklisch die Temperatur zusammen mit der Seriennummer als Kolonnen auf dem Bildschirm ausgegeben. Durch Antippen mit dem Lötkolben kann man so feststellen, wo sich welcher Sensor befindet.

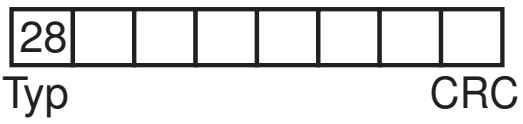


Abbildung 3: Aufbau der Seriennummer

Search ROM — Seriennummer finden

Das Lesen eines ICs anhand der Seriennummer ist simpel: Der Master sendet sie, alle Slaves am Bus vergleichen gegen ihr ROM (Abb.4). Diejenigen, die Unstimmigkeit feststellen, bleiben abgeschaltet und warten auf Reset.

In einer erweiterten Variante dieses Befehls (Abb.5) frägt der Master von allen ICs ihr Bit–0, erst nichtinvertiert

[1] 1–Wire is a device communications bus system designed by Dallas Semiconductor Corp. that provides low–speed data, signaling, and power over a single conductor. (en.wikipedia)

[2] GND = Abkürzung für engl. ground

[3] µSOP — z.B. als DS18B20**U** [1]

[4] NC = not connected

[5] Der Autor verwendet den (altmodischen) Acryllack von Revell, weil dieser gut deckt.

[6] The Dallas 1–Wire network is physically implemented as an open drain master device connected to one or more open drain slaves. (en.wikipedia)

(A) und dann invertiert (B), ab. Da sich in der Verdrahtung immer das Bit–0 durchsetzt[6] (Abb.6), gewinnt man durch die Invertierung weitere Informationen (Tab.1). Als Nächstes entscheidet sich der Master für ein Bit an dieser Position und sendet dieses (C). Damit deselektiert er alle ICs, die damit nicht übereinstimmen. Diese SN@–Sequenz wird für jedes der 64 Bit ausgeführt.

Mit geeignetem Algorithmus ist danach die Seriennummer für ein IC identifiziert und ein Flag gibt an, ob weitere ICs bestimmt werden müssen.

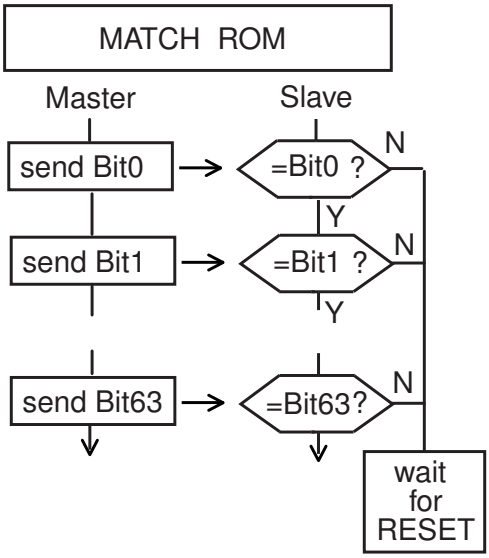


Abbildung 4: Lesen eines Sensors anhand seiner Seriennummer

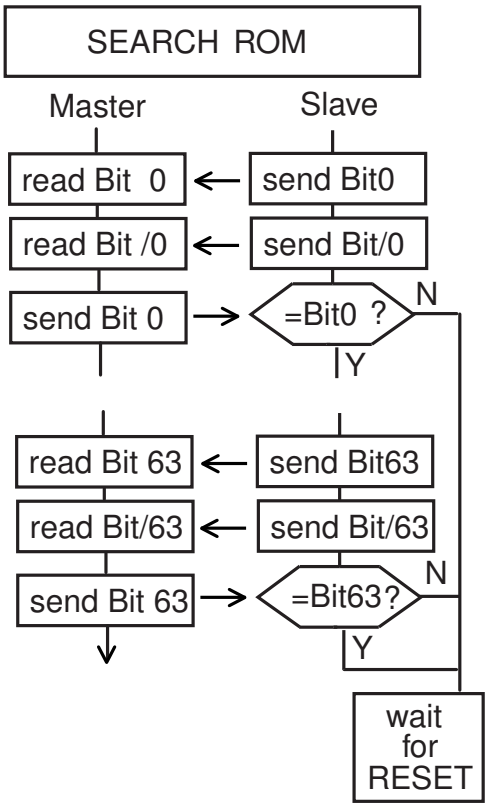


Abbildung 5: Suchbefehl, der die Seriennummer liest

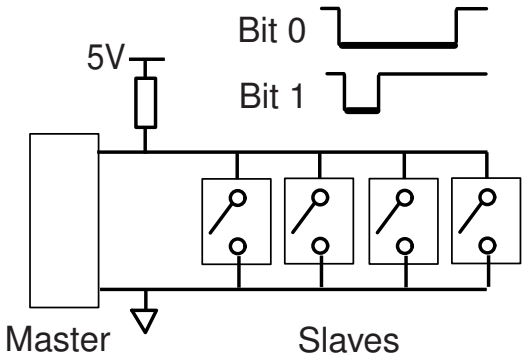


Abbildung 6: Verdrahtungsprinzip

A	B	Antwort–Bit der ICs	Bedeutung
0	0	unterschiedlich	Diskrepanz
0	1	alle 0	Bit ist 0
1	0	alle 1	Bit ist 1
1	1	keine ICs am Bus	Error

Tabelle 1: Auswertung Doppelbit

Algorithmus

Der vom Hersteller angegebene C–Code mit Flussdiagramm ist kompakt, aber undurchsichtig. Das wird allerdings auch nicht viel besser, wenn man ihn nochmal programmiert (Abb.7).

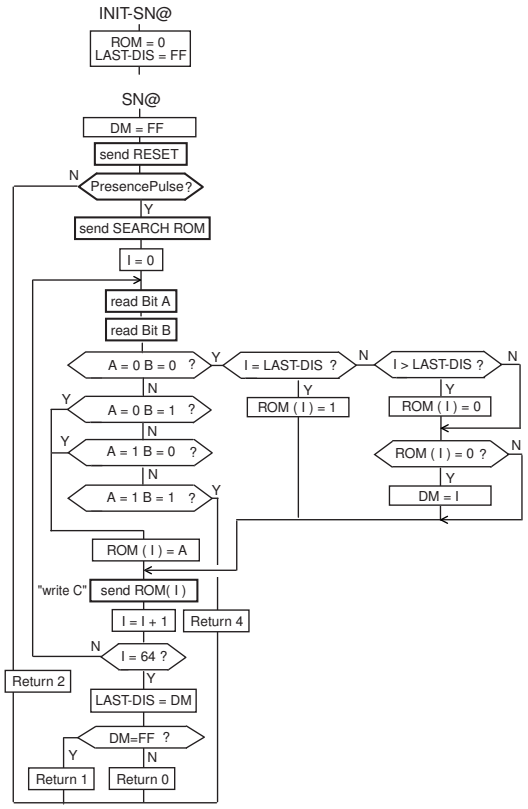


Abbildung 7: Flussdiagramm

Ein Testlauf mit 4 Slaves, die auf 6 Bit verkürzte Seriennummern haben, kann leicht von Hand simuliert werden (Abb.8).

Wenn man nach Lesen von A und B ein Bit sicher identifizieren kann, also Code 01 und 10, dann nimmt man diesen Pegel für C und ändert das Bit im ROM entsprechend. Bei einer Diskrepanz, also Code 00, sind sowohl ICs mit 0 also auch mit 1 an dieser Bit–Position am Bus. Im ersten Durchlauf entscheidet man sich für das entsprechende Bit aus dem ROM. Das ist dann 0, weil dieses komplett auf Null initialisiert worden war. Da wenig ICs am Bus sind, wird recht schnell ein einzelnes isoliert. Bei dem sind dann alle Bits bequem lesbar, weil nun keine Diskrepanzen mehr auftreten können.

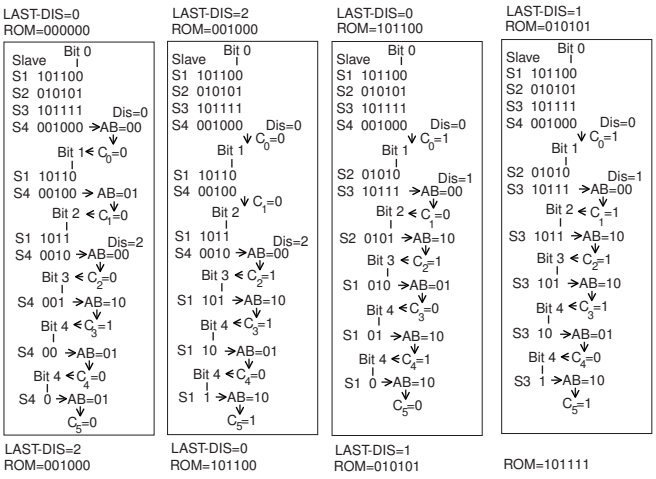


Abbildung 8: Testdaten mit 6 Bit ROM

Diskrepanzen

Nachdem man die erste Seriennummer gefunden hat, muss man verhindern, dass man sie nochmal findet. Deshalb wird bei jedem neuen Suchlauf die alte Seriennummer, d.h. ROM–Bits, als Referenz verwendet.

Diskrepanzen bedeuten immer, dass erst mit Bit–0 und dann nochmal mit Bit–1 gesucht werden muss. Also Verzweigungen in einer Baumstruktur (Abb.9). Man hält sich normalerweise im Baum links, also den mit 0 vorinitalisierten Bits. Möglichst weit unten wechselt man die Richtung nach rechts, toggelt also ein Bit auf 1. Dafür wird das Bit im ROM von 0 auf 1 gesetzt und dieses Bit auch gesendet. Nach dem Toggeln muss man die folgenden Bits im ROM auf 0 setzen.

Toggeln tritt entweder gar nicht oder maximal an einer Verzweigung pro Durchlauf auf. Man weiß bereits, wo: Das wird im vorhergehenden Durchlauf bestimmt. Dazu wird jede Diskrepanz, bei der das Bit im ROM 0 ist, in der Variablen DM gespeichert. Nach Ende des Durchlaufs ist das der gesuchte Wert für LAST–DIS (Abb.8).

DM wird mit einem ungültigen Wert, hier FF, vorinitialisiert. Wenn der nach dem Durchlauf immer noch vorhanden ist, wurde keine unaufgelöste Diskrepanz mehr durchlaufen, also sind alle ICs identifiziert.

Test

Wenn die Funktion „by design“ nicht klar und einfach wird, ist besonders üppiger Test nötig. Mit einer Handvoll echten ICs ist das aber nur unzureichend möglich. Eine Alternative ist dann die Simulation der Slaves, ihre Funktion ist ja simpel. Allerdings sind die 64–Bit–Seriennummern nicht praktikabel. Die 6–Bit–Seriennummern sind genauso aussagekräftig, aber bezüglich Rechenzeit und RAM angenehm kompakt. Abergläubisch wurde entsprechend 64 eine gerade Zahl und nicht 5 genommen. Test aller Kombinationen 2^64 war nicht möglich, immer noch zu umfangreich. Aber man konnte gezielt die wichtigsten Fälle durchspielen, auch einen mit 64 Sensoren maximal bestückten Bus.

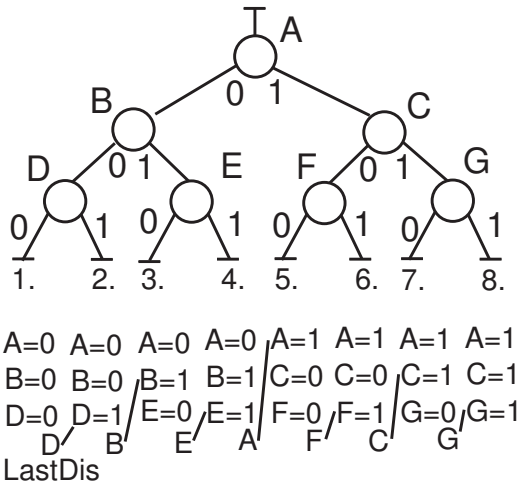


Abbildung 9: Allgemeiner Suchbaum bei Diskrepanzen

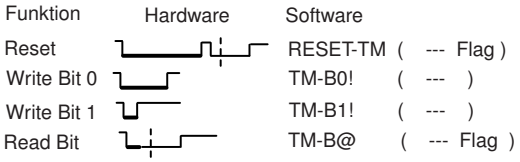


Abbildung 10: Timing — Bedeutung der Pulse

Links

https://en.wikipedia.org/wiki/1-Wire

Literaturhinweise

[1] Datenblatt „DS18B20 Programmable Resolution 1–Wire Digital Thermometer“

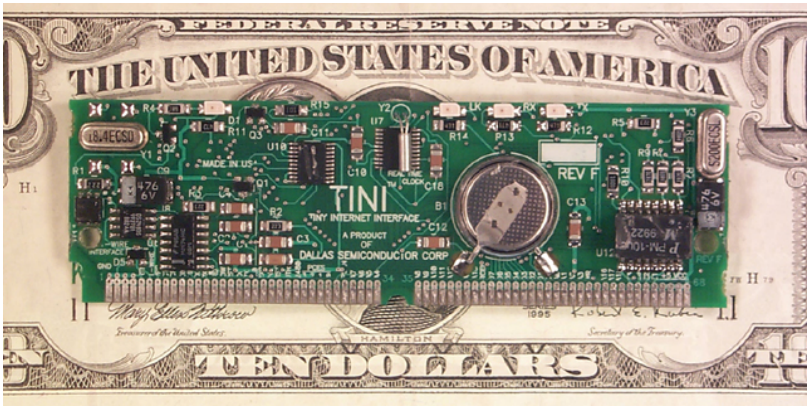
[2] Maxim AN162 „Interfacing the DS18X20/DS1822 1–Wire Temperature Sensor in a Microcontroller Environment“

[3] Maxim AN937 „Book of iButton Standards“

[4] Forth–Magazin „Vierte Dimension“, Heft 2, 2014 (Down–load aus dem PDF–Archiv bei `www.forth-ev.de`)

Listing

```
\ Dallas Bus

\ include:
\ from VD 2014/2 Temperatur-Logger
\ RESET-TM  \ ( --- F ) F = 1 : ok
\ TM-B1!    \ ( --- ) write bit 1
\ TM-B0!    \ ( --- ) write bit 0
\ TM-C!     \ ( UC1 --- ) write byte
\ (TM-C@)   \ ( --- Flag ) read bit

: TM-B@ (TM-C@) ; \ ( --- Flag ) read bit

1 ZVARIABLE LAST-DIS
8 ZVARIABLE ROM  \ 64 bit serial number

: ROM. \ ( --- ) print
7 0 DO ROM I + C@ CH. LOOP CR ;

: (ADR)       \ ( I --- I" addr ) I" = 0 ... 7
DUP 1SHIFT> 1SHIFT> 1SHIFT> ROM +
SWAP B% 111 AND ;

: ROM-B1!  (ADR) B1! ; \ ( I --- ) I = 0...63
: ROM-B0!  (ADR) B0! ; \ ( I --- )
: ROM-B@   (ADR) B@  ; \ ( I --- Flag )

: INIT-SN@  \ ( --- )
ROM 8 00 FILL    FF LAST-DIS C! ;

: SN@
 \ ( --- 0 ) ok: new SN in ROM
 \   --- 1 ) ok: new SN in ROM & all found
 \   --- 2 ) error: no presence pulse
 \   --- 4 ) error: no device code detected
 \   --- 5 ) error: no device code & all found
RESET-TM IF.
F0 TM-C!  \ Search ROM command
FF 0
D% 63 0 DO     \  --- I" flag )
TM-B@          \ read A
IF TM-B@     \ read B
  IF         \ 11 no device code
    4 OR
    1 LEAVE \ write Dummy Bit 1
  ELSE 1     \ 10 Bit 1
  THEN
ELSE
  TM-B@      \ read B
  IF 0       \ 01 Bit 0
  ELSE       \ 00 Discrepancy
    LAST-DIS C@ I =
    IF
      1     \ toggle
    ELSE
      LAST-DIS C@ I U<
      IF   0
      ELSE I ROM-B@
      THEN \  --- I" flag bit )
      IF   1
      ELSE SWAP DROP I SWAP
           0
      THEN
    THEN
  THEN
THEN
IF   TM-B1! I ROM-B1!
ELSE TM-B0! I ROM-B0! THEN
LOOP ELSE 0 2 THEN
SWAP DUP LAST-DIS C! 80 AND IF 1 OR THEN
;

: LIST  \ (  ---  )
       \ list SNs of all devices on bus
INIT-SN@
BEGIN
SN@ DUP CH. SPACE SPACE ROM.
UNTIL ;

\ ----------------------------------------

: MATCH-1  \ ( --- ) serial number in ROM
55 TM-C!   \ Match-ROM Command
7 0 DO ROM I + C@ TM-C! LOOP ;

: S1M      \ (  ---  ) read temperature
RESET-1 MATCH-1 TEMP-1
RESET-1 MATCH-1 SP-1 ;

\  UN1  UN2  UN3  UN4
\  hhhh hhhh hhhh hhhh
\ crc 5  4 3 2  1 0 28

: ROM@ \ ( --- UN1 UN2 UN3 UN4 )
ROM 7 + C@ 8<SHIFT ROM 6 + C@ OR
ROM 5 + C@ 8<SHIFT ROM 4 + C@ OR
ROM 3 + C@ 8<SHIFT ROM 2+  C@ OR
ROM 1+  C@ 8<SHIFT ROM     C@ OR ;

: ROM! \ ( UN1 UN2 UN3 UN4 ---  )
DUP ROM   + C! 8SHIFT> ROM 1+  C!
DUP ROM 2+  C! 8SHIFT> ROM 3 + C!
DUP ROM 4 + C! 8SHIFT> ROM 5 + C!
DUP ROM 6 + C! 8SHIFT> ROM 7 + C! ;
```



Clock Works 2 — Anzeige a la Abakus

Erich Wälde

Im ersten Teil dieser Reihe [1] ging es um die Programm–Bausteine zur Realisierung einer Uhr. In diesem Teil soll eine alternative Anzeige vorgestellt werden. Die einzelnen Ziffern werden mit 5– und 1–wertigen LEDs dargestellt, wie die Kugeln beim Abakus. Eine Idee aus der Kategorie Yes, we can!

Abakus?

Ein Abakus (Zählrahmen)? War das nicht das lustige Holzrähmchen mit den Holzkugeln auf Stäbchen? Hatte nicht jeder von uns irgendwann mal so ein Ding in seinen unschuldigen Kinderhänden und wusste nicht, dass man damit *höhere* und auch ganz *praktische* Mathematik machen kann? Das zweite Mal begegnete mir das Ding in einem Reiseführer über Russland. Sinngemäß: man möge sich nicht wundern, wenn nach dem Kurbeln der mechanischen Registrierkasse der Verkäufer einen Abakus zur Hand nimmt, um das Resultat mit geübten, flinken Fingern zu prüfen. Aha.

Das dritte Mal begegnete mir das Ding in der Vorlesung über *Rechnerarchitektur* (ca. 1992 in Heidelberg). Sapperlott! Und ja, diesen Teil vom Skript habe ich heute noch. *Anleitung für die chinesische Rechenmaschine Abacus* — ein Heftchen mit ca. 25 Seiten. Den Abakus korrekt bedienen habe ich leider nicht gelernt, aber ich habe verstanden, dass die Kugeln auf einem Stab eine Ziffer darstellen. Es gibt dann zwar verschiedene Möglichkeiten, die minimalste Darstellung findet sich in der japanischen Variante: Auf jedem Stäbchen sind zwei Felder (mit einem Mittelrahmenstück abgeteilt). Im schmalen Feld befindet sich eine Kugel. Ist diese Kugel zur Außenseite geschoben, zählt sie nichts. Ist diese Kugel zur Mitte geschoben, so zählt sie fünf. Im breiten Feld befinden sich vier Kugeln. Jede Kugel, die nach außen geschoben ist, zählt nichts, jede Kugel, die zur Mitte geschoben ist, zählt eins. Mit den vier Kugeln kann man also die Werte 0 bis 4 darstellen, zusammen mit der Kugel im schmalen Feld die Werte 0 bis 9. Im Dezimalsystem ist das ausreichend, um die für uns so gewohnten Ziffern der Werte 0 bis 9 darzustellen, jedes Stäbchen bedeutet also eine Ziffer.

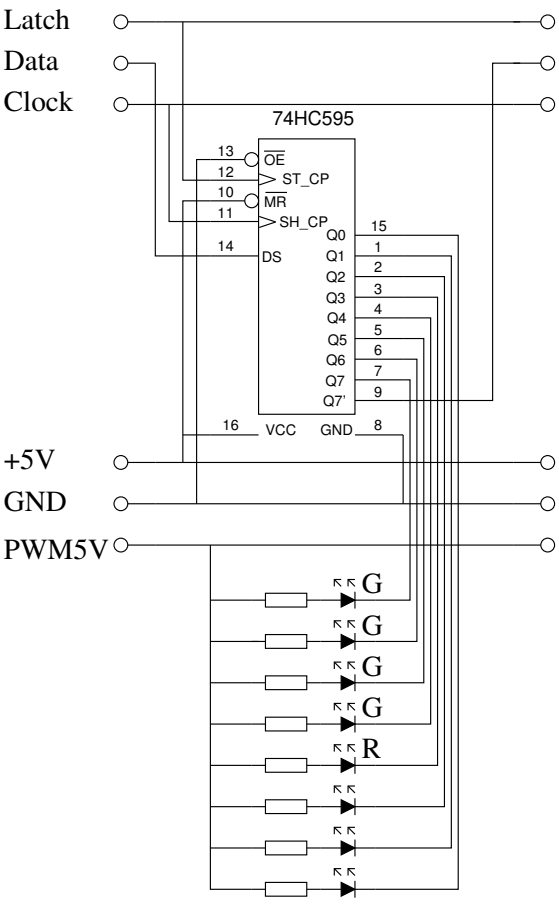
Wikipedia [3] sagt, dass das Rechnen mit dem Abakus eine so wichtige Kulturtechnik sei, dass sie 2013 zur Liste der immateriellen Kulturerbe hinzugefügt wurde. Auf so einer Liste findet sich auch das Morsen!

Abakus!

Die Darstellung der Ziffern lässt sich recht einfach auf LEDs übertragen: eine rote LED, die 5 zählt, wenn sie leuchtet. Vier gelbe LEDs, die jeweils 1 zählen, wenn sie leuchten. Man braucht also für jede darzustellende Ziffer fünf LEDs — das sind zwei weniger, als bei den sog. 7–Segment–Anzeigen. Zwar können diese auch noch einen Dezimalpunkt und mit etwas Fantasie die Ziffern A bis F darstellen, aber ein Hexadezimalsystem brauchte ich erst mal nicht.

Da ich ein expliziter Anhänger von für mich flimmerfreien Anzeigen bin, habe ich also auf Streifenraster eine Anzeige realisiert, die 3 Paare von Ziffern besitzt. Jede Ziffer wird über ein Schieberegister angesteuert. Damit kann ich erst einmal die Daten für alle Ziffern übertragen und anschließend die übertragenen Werte gleichzeitig auf die Ausgänge schalten (latch). Will man die Helligkeit der LEDs steuern, dann benötigt man eine mit höherer Frequenz PWM–förmig geschaltete Stromversorgung für die LEDs. Das ist zwar aufwendig aber gut für die Augen. Dafür kann man die Anzeige mit drei Leitungen steuern (`data`, `clock`, `latch`) mit einer vierten Leitung die Helligkeit steuern (`pwm`) und sie sieht flimmerfrei aus, wenn die PWM–Frequenz hoch genug liegt ($> 100\,\mathrm{Hz}$).

Schaltung



Die Schaltung besteht aus einem Schieberegister (74HC585), an welches bis zu acht LEDs angeschlossen

werden können. Dargestellt ist lediglich eine Ziffer, für jede weitere Ziffer wird eine identische Einheit angereiht. Dabei ist nur zu beachten, dass die Daten über DS (14) in das Schieberegister gelangen. Nach 8 Bits erscheinen die Werte am Ausgang Q7' (9), welcher mit dem Pin DS des nachfolgenden Schieberegisters verbunden wird.

Aus löttechnischen Gründen wurden die vier gelben LEDs an die höherwertigen Bits (Q4...Q7) angeschlossen. Die drei unbeschrifteten LEDs (Q0...Q2) sind in meiner Anzeige nicht bestückt. Bestückt man die fehlenden LEDs, dann erhält man eine Binäranzeige — im Programm muss dann die Ausgabe der Ziffern angepasst werden.

Im einfachsten Fall wird die Stromversorgung der LEDs (`PWM5V`) direkt mit dem Pin `+5V` verbunden. Will man die Helligkeit aber steuern können, dann muss ein PWM–Signal (Puls–Weite–Modulation) die LEDs steuern. Das kann man am einfachsten mit einem MOSFET erreichen (Gate an den steuernden PWM vom Kontroller, Drain an `+5V`, Source an `PWM5V`). Normalerweise reicht ein MOSFET für die ganze Anzeige, man kann diesen Baustein aber auch bei jeder Ziffer bestücken.

Das Programm

Bis auf die Steuerung der Anzeige ist das Programm identisch zu dem im letzten Teil vorgestellten Programm. Es sollen hier also nur die neuen Teile vorgestellt werden.

main: Pins, Init

```
\ abakus display
PORTD 2 portpin: sr_latch
PORTD 3 portpin: sr_clock
PORTD 4 portpin: sr_data

\ make redirects return to console
' emit defer@ constant emit.orig
: >con  emit.orig to emit ;

\ --- clock display (deferred)
include lib-avr8/forth2012/core-ext/
        avr-defers.frt
include ewlib/clock_display_defer.fs
include ewlib/clock_display_pollin_2x8.fs
                \ +clock.display.pollin_2x8

include ewlib/clock_display_abakus_1.fs
                \ +clock.display.abakus1

: init
  \ ...
  -clock.display
  +clock.display.abakus1
  cd.time
;
```

Übertragung an das Schieberegister

Das Schieberegister 74HC595 ist ein Baustein mit drei Steuereingängen (clock (`SH_CP`), data (`DS`), latch oder store (`ST_CP`)), 8 Ausgängen (`Q0` ... `Q7`) sowie einem weiteren Ausgang (data output, `Q7'`). Dieses Signal wird mit data des nächstes Bausteins verbunden, so dass die Daten weitergereicht werden können. Schiebt man in zwei so verbundene Schieberegister zwei Zeichen, dann wird das zuerst gesendete Zeichen auf dem zweiten Schieberegister landen, und das zweite gesendete Zeichen auf dem ersten Schieberegister.

Das Schieberegister wird mit altmodischer aber bewährter Bit–Schieberei angesprochen. Zunächst werden die zugehörigen Pins am Kontroller initialisiert.

```
\ needs pin definitions
\     PORTD 2 portpin: sr_latch
\     PORTD 3 portpin: sr_clock
\     PORTD 4 portpin: sr_data

: +sr
  sr_latch pin_output  sr_latch high
  sr_clock pin_output  sr_clock low
  sr_data  pin_output  sr_data  high
;
```

Ein einzelnes Bit wird auf der Datenleitung angelegt, danach durch einen Impuls auf der Clockleitung für gültig erklärt und vom Schieberegister übernommen. Bei jedem Clock–Impuls werden die vorhandenen Daten um eine Position weitergeschoben.

```
: bit>sr ( bit -- )
  if sr_data high else sr_data low then
  sr_clock high  sr_clock low
;

: get.bit ( byte pos -- bit )
  1 swap lshift    \ -- byte bitmask
  and              \ -- bit
;
```

Ein ganzes Byte wird mit dem höchstwertigen Bit zuerst an das Schieberegister gesendet. Wahrscheinlich könnte man diese Reihenfolge hier auch umdrehen, sollte man die Anzeige *auf dem Kopf* betreiben — eine weitere Stelle zum Schrauben.

```
\ clock one byte out, MSB first!
: byte>sr ( byte -- )
  0 7 do
    dup i get.bit
    bit>sr
  -1 +loop
  drop
;
```

Die LEDs sind von der Versorgungsspannung her angeschlossen. Damit die LED leuchtet, muss an den zugehörigen Ausgang eine Null geschrieben werden. Deswegen habe ich vor dem Senden ein `invert` spendiert, was eine Kopfverrenkung spart.

```
: >sr
  invert
  byte>sr
  sr_latch low sr_latch high
;
```

Um nun eine gültige Anzeige für einen bestimmten Wert zu erreichen, werden die Werte (0...9) als Index in eine kleine Tabelle benutzt. In der Tabelle steht das zugehörige Bitmuster, um auf der Anzeige das gewünschte Symbol zu erhalten. Beim Verdrahten der Ausgänge `Q0 ... Q7` habe ich eine für die Verkabelung angenehme Belegung verwendet. Das führt dazu, dass die Muster für die Anzeige der Ziffern etwas umständlich ausfallen. Diese Tabelle ist gegebenenfalls an die konkrete Anzeige anzupassen.

```
create AbakusDigits
$00 , \ 0
$10 , \ 1
$30 , \ 2
$70 , \ 3
$F0 , \ 4
$08 , \ 5
$18 , \ 6
$38 , \ 7
$78 , \ 8
$F8 , \ 9
```

Bleibt nur noch die Ausgabe eines kompletten Datensatzes zu bewerkstelligen. Die Werte werden der Reihe nach an die Kette der Schieberegister durchgegeben. Hier mit dem niedrigsten Wert, der Einer–Sekunde, zuerst, denn das letzte Schieberegister in der Kette zeigt diese an. Dreht man das Display auf den Kopf, ändert sich auch diese Reihenfolge. Nachdem alle Werte übertragen sind, werden sie zur Anzeige gebracht (latch).

```
\ transfer n digits
: type.sr ( xn-1 .. x0 n -- )
  0 ?do
    dup 0 #10 within if
    else
      drop 0
    then
    AbakusDigits + @i  invert  byte>sr
  loop
  sr_latch low sr_latch high
;
```

Anzeige!

Die Ansteuerung der Abakus–Anzeige wird über drei Funktionen realisiert, genauso wie die Ansteuerung LCD–Anzeige im ersten Teil. Da die vorliegende Anzeige das Datum nicht anzeigt, bleibt die `.date`–Funktion leer. Die `.time`–Funktion besorgt die aktuellen Werte, zerlegt sie in Zehner und Einer Anteile und überträgt diese Werte dann an die Kette der Schieberegister. Die dritte Funktion registriert die `.time`–Funktion in `cd.time` — das ist die Funktion, die vom übergeordneten Programmteil aufgerufen wird. Selbstverständlich kann man auch `clock.display.abakus1.time` direkt aufrufen.

```
: clock.display.abakus1.date   ( -- )
;
: clock.display.abakus1.time   ( -- )
  hour @ #10 /mod swap
  min @  #10 /mod swap
  sec @  #10 /mod swap
  6 type.sr
;
: +clock.display.abakus1
  ['] clock.display.abakus1.time    to cd.time
;
```

Aktivieren der Anzeige

Die Anzeige muss jetzt beim Sekundenwechsel beschrieben werden. Mit der in `init` gezeigten Intitalisierung erledigt `cd.time` die Angelegenheit.

```
: job.sec
  \ ...
  cd.time
  \ alternately
  \ clock.display.abakus1.time
;
```

Schluß

Diese Uhr ist ebenso benutzbar, wie die im ersten Teil vorgestellte, lediglich die Anzeige ist anders. Und diese hat eine funktionierende serielle *Konsole* für den erhöhten geek–Faktor. Es sind etliche Uhren im Handel, die mit binären Anzeigen arbeiten ([4,5]). Eine *Zählrahmen*–Anzeige ist mir bislang aber noch nicht begegnet. Wahrscheinlich kann man das mit einer Uhr wie der pebble ([6]) realisieren, möglicherweise gibt's das schon.

Mit der hier gezeigten Technik sind jede Menge weitere Anzeigen möglich. Man kann durch andere Anordnung der LEDs (je 8 pro Schieberegister, in zwei Gruppen zu je drei *Ziffern* auch eine binäre Anzeige mit Datum realisieren. Die Ansteuerung bleibt gleich, die Umrechnung von Zahl in Bitmuster vereinfacht sich hoffentlich. Ok, genau genommen braucht man dann für das Jahr noch ein paar Bits mehr, also eine weitere Ziffer. Es sei denn man zeigt Jahr−2000 an.

Oder 60 LEDs im Kreis mit 8 Schieberegistern je 8 Bit angesteuert, für eine quasi analoge Minutenanzeige. Ähnlich mit 12 LEDs im Kreis für eine analoge Stundenanzeige.

Eine binäre Anzeige mit 32 Bit, welche die Epochensekunden anzeigt — dann sieht man das Jahr–2038–Problem ([7]) besser kommen.

Man kann durch 7–Segment–Anzeigen anstelle einzelner LEDs eine Ziffernanzeige realisieren. Wenn man die einzelnen Ziffern groß genug wählt und das Layout *anreihbar* gestaltet, dann kann man auch längere Anzeigen, z.B. der Epochensekunden, realisieren. Und das bei moderaten Änderungen am Programm.

So Sachen

1. Ich wurde nach dem *Schaltplan* der Uhr im ersten Teil gefragt. Meine Antwort: der wohnt in den Pin–Definitionen am Anfang des Programms.
2. Wenn die konkrete Anordung der Pins zum Löten optimiert ist, dann muss man das in der Software kaschieren (Bit–Reihenfolge in der Anzeige einer Ziffer). Die gute Nachricht: man *kann* das sogar machen! Wir Softwerker müssen in unserem Programmierleben oft

die Eigenheiten oder verdrehten Anschlüsse der Technik in Software ausgleichen.
3. Bei einer Kette von Schieberegistern muss für jede neue Anzeige der komplette Datensatz übertragen werden. Bei der LCD–Anzeige im ersten Teil konnte man gezielt etwa nur die Sekunden neu anzeigen.
4. Die Uhr im ersten Teil hatte *Zicken*. Manchmal hat sie auf der Kommandozeile Mist ausgerechnet. Also z.B. waren `60 3 * 21 +` nicht immer `201`, sondern nur meistens. Irgendwann ist mir aufgefallen, dass ein falsches Ergebnis zwar mit `.` angezeigt wird, aber die Ausgabe von `.s` immer korrekt aussah. Es wurde also nicht falsch gerechnet, sondern falsch angezeigt. Danach war's einfach: Der Hintergrund–Task beschreibt die LCD–Anzeige ganz forthig per Ausgabeumleitung. Dort wurde also ebenfalls der Befehl `.` benutzt, der dann auf `<# # #>` zurückgreift. Gibt man am prompt gleichzeitig Ziffern aus, dann kommen sich die beiden Tasks durch die gemeinsam genutzte *hold area* in die Quere. Das ist `AmForth`–spezifisch (Version 6.3) und auch noch nicht geflickt, es ist jetzt nur klar, wo das herkommt. Bei der Abakus–Anzeige existiert dieses Problem nicht, da werden die Ziffern *von Hand* formatiert und ausgegeben.

Verweise

1. VD 2016-04, S.15ff — E. Wälde, Clockworks 1 Die kleine Uhr
2. Film vom Dt. Museum `http://www.youtube.com/watch?v=jwabVzlobZI`
3. `https://de.wikipedia.org/wiki/Abakus_%28Rechenhilfsmittel%29`
4. `https://www.getdigital.de/binaerearmbanduhr.html`
5. `https://www.getdigital.de/binary-wall-clock.html`
6. `https://de.wikipedia.org/wiki/Pebble_%28Smartwatch%29`
7. `https://de.wikipedia.org/wiki/Jahr-2038-Problem`

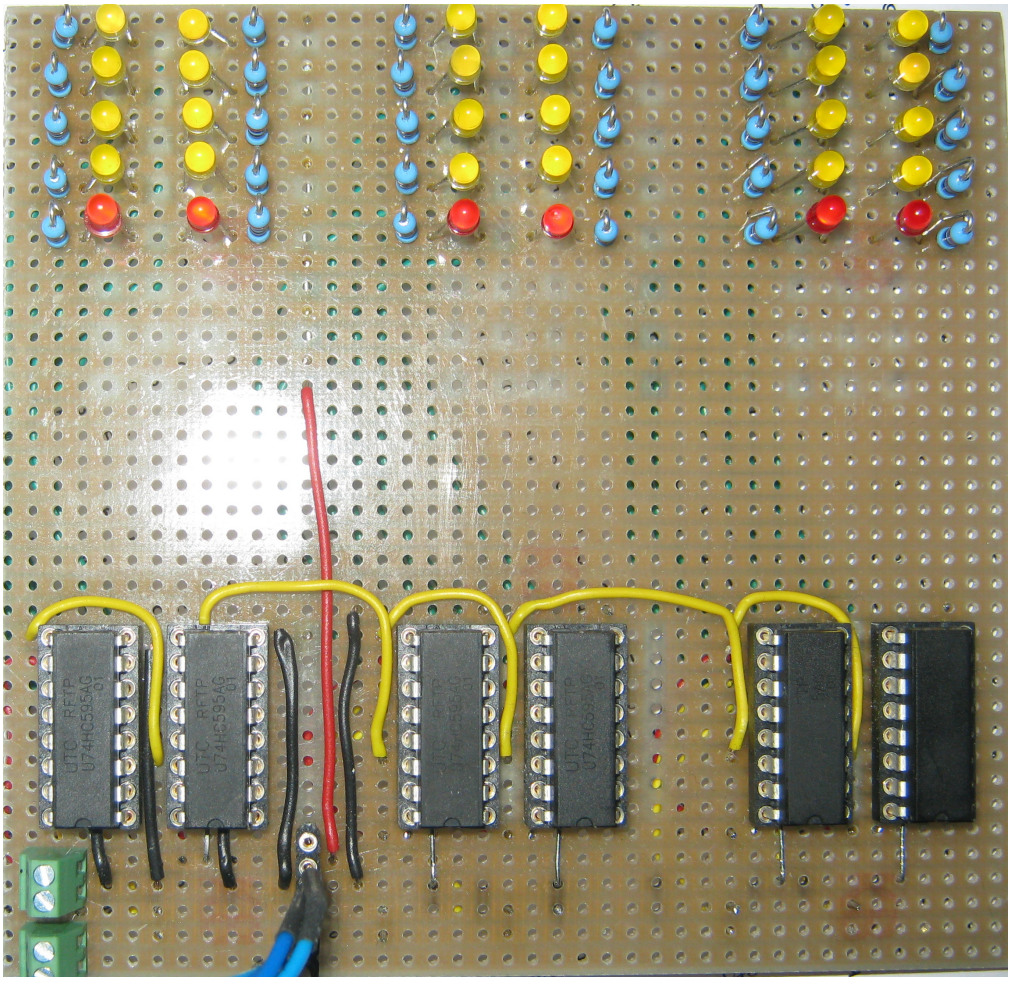


Abbildung 1: Aufbau der Abakus–Anzeige auf Lochraster–Platine. Die gelben Kabelverbindungen leiten die Daten von einem Schieberegister zum nächsten.

A Compilation On Demand Mechanism

Klaus Schleisiek

µController haben meistens kleine Programmspeicher, so dass das Einbinden sämtlicher gängigen Forth–Worte signifikante Anteile des vorhandenen Platzes belegen. Das ist viel toter Code, wenn nur wenige dieser Worte in der konkreten Anwendung benutzt werden. Wie schön wäre es, einen Lademechanismus zu haben, der nur die Worte lädt, die die Anwendung tatsächlich braucht. Einen solchen Mechanismus habe ich auf gforth 0.6.2 für den Microcore–Crosscompiler (µCore) entwickelt.

Die Umgebung

Im realen Einsatz läuft auf einem Hostrechner ein Crosscompiler, der Code für einen µController, das Target, erzeugt. Dann können auf dem Host Worte für das Target definieren werden, ohne im Target notwendigerweise sofort Code zu erzeugen. Der hier vorgestellte Referenzcode erzeugt nur Code auf dem Host, bleibt also innerhalb desselben Kontextes. Das ist eigentlich Blödsinn, weil dadurch nur der erzeugte Code aufgebläht wird, aber er realisiert beispielhaft den gesamten Mechanismus, der in den jeweiligen Crosscompiler integriert werden muss.

Der Mechanismus

`lib.fs` ist ein Libraryfile mit den Definitionen oft benutzter Worte. Ein Libraryfile entsteht aus einem ladbaren Sourcefile, in dem an bestimmten Stellen Tilde–Zeichen `~` eingefügt werden. Code, der zwischen aufeinanderfolgenden `~` steht, ist eine *Librarygroup*, die dann, wenn eines der Worte innerhalb der Gruppe benutzt wird, als Gesamtheit ins Target compiliert wird. Dadurch lassen sich Adressberechnungen und die Definition von Konstanten vor oder nach einer :–Definition einbeziehen. Mit `library <libname>` werden für alle Worte, die Einträge im Wörterbuch anlegen — `Create : Constant Variable` — sozusagen in einer Präcompilationsphase, jeweils Libdef–Einträge mit einem Verweis auf den Sourcecode der zugehörigen Librarygroup erzeugt. Im Target landet während der Präcompilation nichts. Wird später der Anwendungscode fürs Target compiliert, dann wird manchmal ein Libdef–Wort gefunden und, weil es *immediate* ist, sofort ausgeführt. Zunächst werden alle für die Kompilation wichtigen Zeiger auf den Zustand vor Beginn des aktuellen `:` zurückgesetzt — auch der Zeiger auf die nächste freie Stelle des Programmspeichers im Target. Dann wird der in der Präcompilationsphase gespeicherte Kontext der jeweiligen Librarygroup wiederhergestellt und dort der Sourcecode bis zur folgenden `~` geladen, so dass Code im Target erzeugt wird. Gleichzeitig wird dadurch das gerade ausgeführte Libdef–Wort redefiniert und künftig statt dessen die Referenz auf den soeben kompilierten Code gefunden. Danach wird die Kompilation der Anwendung am Anfang des abgebrochenen `:` wieder aufgenommen. Der Mechanismus ist rekursiv und auch beim Laden von Libdefs dürfen weitere Libdefs vorkommen. Libraries können also Libdefs aus sich selber oder anderen Libraries nachladen.

Die Testanwendung

`library.fs` ist der *compilation on demand* Mechanismus und wird später im Crosscompiler integriert.

`lib.fs` ist die „Bibliothek“.

```
cr .( pre-compiling lib.fs )
~ : dudel ." didel " .( 1 ) daddel @ . .( 2 ) ;
~ 2 Constant Konstant Variable daddel Konstant daddel !
~ Variable resource 2 resource !
~ : rumpel ( -- ) ." alle " resource @ . ;
```

`test.fs` ist das Loadfile der Anwendung und während des Debuggens entstanden. Nachdem der Bibliotheksmechanismus geladen und eine Bibliothek präcompiliert wurde, wird mit den folgenden drei :–Definitionen der gesamte Mechanismus getestet. Während des Ladens werden kurze Meldungen ausgegeben, so dass der komplexe Ablauf nachvollzogen werden kann:

```
: drebel   [ cr ] .( a ) dudel ." hei " ;
: daudel   ." ta" dudel ." dei " daddel .( b ) @ . ;
: doedel   .( c ) rumpel  .( d )  1 resource +! ;
cr .( finished compiling, words in lib Vocabulary: ) words
cr drebel cr daudel cr doedel cr doedel
```

Ein `include test.fs` erzeugt folgende Meldungen:

```
pre-compiling lib.fs
a 1 redefined daddel  1 2 redefined dudel
a b c redefined resource  redefined rumpel  c d
finished compiling, words in lib Vocabulary:
rumpel resource dudel daddel Konstant rumpel
resource daddel dudel
didel 2 hei
tadidel 2 dei 2
alle 2
alle 3  ok
```

Was passiert da? `pre-compiling lib.fs` wird ausgegeben, weil wir `library lib.fs` ausgeführt haben. Dann ein `cr` und das `a` beim Laden von `drebel`. `dudel` gibt es nur als Predef und `drebel` wird wieder verworfen. Die `1` verdanken wir der Kompilation von `dudel` aus der Bibliothek. Dann jedoch ist `daddel` auch nur ein Predef, `dudel` wird verworfen und erstmal `Konstant` und `daddel` in der darauffolgenden Libgroup geladen. Das sehen wir an `redefined daddel` und fortan wird die Variable als echter Code und nicht mehr als Predef gefunden.

Nun kann die Kompilation von `dudel` wiederholt werden, was uns nicht nur `1`, sondern auch `2` beschert, weil `daddel` ja gerade geladen worden ist. `redefined dudel`

schließt die Definition von `dudel` ab. Die Predef `dudel` hat ausgedient.

Das `cr` gefolgt von `a` und `b` zeigt, dass `daudel` jetzt vollständig geladen werden konnte. Die Kompilation von `doedel` beginnt und beschert uns das `c`. `rumpel` ist aber wieder eine Predef. Die Libgroup von `rumpel` wird geladen, das scheitert aber an der Libdef resource, so dass auch `rumpel` abgebrochen wird, um erstmal `resouce` zu laden, was uns `redefined resource` beschert. Dann geht's zurück zu `rumpel` und wir sehen an `redefined rumpel`, dass `rumpel` diesmal vollständig geladen werden konnte.

Zurück im File `test.fs` kann endlich `doedel` stolperfrei kompiliert werden und wir bekommen `c d` angezeigt. In den Folgezeilen wird das Vocabular `lib` angezeigt und der compilierte Code getestet.

Die Realisierung

Dazu gehen wir das file `library.fs` Schritt für Schritt durch:

Die Variable `Libfile` ist ein Pointer auf den vollständigen Filenamen (incl. Path) der Bibliothek, die mit dem Befehl `library <libname>` gerade präkompiliert wird.

Die Variable `Libgroup` ist ein Pointer auf eine Datenstruktur, um den Kontext einer Libgroup zu charakterisieren. Diese enhält hintereinander die folgenden Elemente:

1. Ein Pointer auf den Filenamen der Bibliothek (`Libfile`)
2. Die Byteposition im Bibliotheksfile unmittelbar hinter der `~`, mit der eine Libgroup beginnt.
3. Die aktuelle Kompilationswortliste (`Current`)
4. Der aktuelle Kompilationskontext (`get-order`).

Mit diesen Informationen kann später der gesamte Kompilationskontext wiederhergestellt werden, wenn eine Libgroup ins Target geladen wird, weil ein Worte daraus von der Anwendung gebraucht wird.

Die Variable `Libload` wird auf `true` gesetzt, wenn eine Libgroup tatsächlich geladen wird. Sie wird als State–Information bei der Definition von `~` gebraucht.

Mit der Variablen `Libdepth` wird überprüft, ob bei der Präcompilation einer Bibliothek Elemente auf dem Stack zurückgeblieben sind.

`source-position` errechnet die Fileposition, die dem aktuellen Zustand von `>in` entspricht. Das Wort ist systemabhängig und in Standard–Forth nicht realisierbar.

`libgroup,` legt die Datenstruktur einer Libgroup an.

`>libgroup` stellt den Kontext einer Libgroup wieder her, so dass sie geladen werden kann.

`load-libgroup` lädt eine Libgroup, wobei der gerade aktuelle Kompilationskontext gesichert und am Ende wiederhergestellt und fortgesetzt wird.

Bei `Tmarker` wird zu Beginn jeder :–Definition mit `mark-target` die Position `here` des nächsten freien Platzes des Programmspeichers, die Tiefe `depth` des Datenstacks und der Zustand des Eingangsdatenstroms abgelegt, so dass jede begonnene :–Definition mit `restore-target` verworfen und später erneut aufgenommen werden kann. `mark-target` stellt `>in` um zwei Bytes zurück, so dass bei der späteren Re–interpretation wieder mit dem abgebrochenen `:` begonnen wird.

In die Wortliste `Predefined` werden alle Worte kompiliert, die während der Präcompilation gebraucht werden. Dies sind außer den definierenden Worten auch Worte für die bedingte Kompilation `wie [IF]` und Ähnliche.

Danach folgen zwei `:noname`–Definitionen, deren xt als Konstante benutzt werden. Dadurch werden die Prädefinitionen (Predefs) typisiert, um z.B. `Variable` auch während der Präcompilation interpretieren zu können.

Mit `Libpredef` wird ein Wort in der Predefined Wortliste kreiert. Diese verhalten sich später typgemäß.

`Libdef` erzeugt während der Präcompilation die immediate Worteinträge in der aktuellen `current` Kompilationswortliste, die bei Ausführung den Quellcode „ihrer" Libgroup nachladen.

`Libcreate` sodann erzeugt beides, eine `Libdef` und eine `Libpredef` und wird benutzt, um zB `Variable` und `Create` für die Präcompilation zu realisieren.

Jetzt kommen wir endlich zum Zentrum des *compilation on demand* Mechanismus, dem *pre–compiler*. `addr` und `u` verweisen auf einen im Quellcode geparsten Namen.[1] Dann wird die Wortliste Predefined durchsucht und gefundene Worte ausgeführt. Zahlen werden auch wie üblich verarbeitet und ansonsten wird der Name sang- und klanglos verworfen.

Deshalb können bereits während der Präkompilation Konstanten und Variable verarbeitet werden, um z.B. `[IF]` zu füttern. Mir ist allerdings nicht klar, ob diese Möglichkeiten auch wirklich gebraucht werden, oder ob man nicht immer zum nächsten Token übergehen kann, wenn in `Predefined` nichts gefunden wurde.

`library` wird benutzt wie `include` und startet die Präkompilation einer ganzen Bibliothek.

Die Semantik von `:` wird erweitert und künftig immer zuerst ein `mark-target` gemacht.

Für die Definition von `~` brauchen wir die Zustandsvariable `Libload`. Im normalen „Forthmodus" tut `~` nichts, während der Präcompilation jedoch wird an dieser Stelle der Ladeprozess der vorherigen Libgroup beendet, indem im Sourcefile ans Ende gesprungen wird.

`scan-for-;` überspringt bei der Präcompilation von `:` den Sourcecode bis zum folgenden `;` .

[1] Die erste Codezeile von pre–compiler ist ein gforth–Bugfix, um [IF] usw. nutzen zu können.

Wir kommen nun zu den Worten, die während der Präkompilation ausgeführt werden können. Das ist erstmal eine anwendungsspezifische Liste von Alias–Definitionen, um existierende Worte auch während der Präkompilation zugänglich zu machen.

~ überprüft die Stackintegrität und legt dann die nächste Libgroup an.

`Constant` erweitert den Wortschatz in `Predefined` .

`Create` und `Variable` werden sowohl als `Predef` als auch in `Predefined` als Eintrag vom Typ `#var` angelegt.

Die Predefined Worte schließen mit `:` ab. Es wird sowohl ein `Predef` als auch ein Eintrag in Predefined vom Typ `#:` erzeugt und der folgende Quelltext bis zum nächsten `;` übersprungen.

Dann folgen noch ein paar Worte, die ich während der Entwicklung ständig zum debuggen benutzt habe.

Systemabhängigkeit des Codes

Ulrich Hoffmann hat den Code daraufhin durchgesehen, ob er auch mit Standard–Forth–Worten realisiert werden könnte. Kurze Antwort: Nein. Noch nicht einmal generell gforth–kompatibel ist der Code, weil aus gforth_0.6.2 `Loadfilename` benutzt wird, das in gforth_0.7.9 gar nicht mehr existiert, weil der Sourceverwaltungsmechanismus grundlegend geändert wurde. Und dank der neu eingebauten Recognizer gibt es in 0.7.9 auch Parser nicht mehr. Seis drum, der Referenzcode läuft auf gforth_0.6.2 — was auf Windows_10 leider nicht mehr installierbar ist.

Im Standard fehlt sowohl eine Möglichkeit, den äußeren Interpreter temporär zu ändern (`pre-compile`), eine :–Definition abzubrechen und den Systemzustand vor Beginn der :–Definition wiederherzustellen, als auch die Möglichkeit, den Zustand der Sourcefileumgebung zu sichern, ein Libraryfile zu öffnen und daraus zu compilieren und danach wieder zur vorherigen Sourcefile–Umgebung zurückzukehren. Die eigentlich dafür vorgesehenen Worte `save-input` und `restore-input` erlauben nämlich nicht den Wechsel des Files.

Listings

library.fs

```
\
\ Last_change: KS 19.03.2017 16:52:18
\
\ ***************************************************************
\ Compilation on demand, "host only" model implementation.
\
\ Take this model implementation to roll your own cross compiler
\ that compiles dense code for skinny embedded target systems.
\
\ The code loads on aged gforth_0.6.2 and unfortunately, there
\ are a number of system dependent words, which can not easily be
\ replaced by Standard words. Ulrich Hoffmann looked at the code
\ from a Standards point of view and did the easy ones.
\
\ In principle, the Standard is not prepared to deal with
\ relinquishing the compilation of a :-definition, loading code from
\ another file, and then re-loading the previously interrupted
\ :-definition from the first file. In addition, there is no hook
\ into the outer interpreter available in the Standard.
\
\ The system dependency of "marking" and "restoring" a colon
\ definition (mark-target, restore-target) in a real host/target
\ compilation system is less problematic, because you will have
\ to port the code to some concrete cross compiler anyways.
\
\ 18-mar-2017
\ kschleisiek at freenet.de is the original author
\ uho at xlerb.de remarked on standard (in)compatibility
\
\ ***************************************************************

: parse-name  ( -- caddr u )   parse-word ;  \ gforth_0.6.2 misnomer

\ words already present in gforth:
\ : Alias    ( <name> xt -- )   Create , Does> @ execute ;
\ : Root     ( -- )             get-order dup pick rot rot nip set-order ;
\ : wordlist-words ( wid -- )  >r get-order r> 1 set-order words set-order ;
\ : rdrop    ( -- )             postpone r> postpone drop ; immediate
\ : ?EXIT    ( f -- )           IF  rdrop  THEN ;
\ : -rot     ( n1 n2 n3 -- n3 n1 n2 )  rot rot ;

```

```
Base @ hex   Create Prefixes  10 , 2 , A ,   Base !
\                             $   %    &  base prefix character
: ?base ( caddr u -- caddr' u' )
   over c@ [char] $ - dup 4 u<
   IF  cells Prefixes + @   base !  1 /string  EXIT THEN
   drop
;
: ?sign ( caddr u -- caddr' u' flag )
   over c@ [char] - = >r  r@ IF  1 /string  THEN   r>
;
: integer? ( caddr u -- 0 / n -1 / d 0> )
   Base @ -rot   0 >r   ?sign >r   ?base   0. 2swap
   BEGIN  >number ?dup 0=
      IF  drop  rot Base !   r> IF  dnegate  THEN
          r> ?dup ?EXIT      \ double number
          drop True  EXIT    \ single number
      THEN
      over c@ [char] . =
   WHILE  r> rdrop over >r >r   1 /string
   REPEAT
   2drop 2drop rdrop rdrop   Base !  False
;
: cell-   ( n1 -- n2 )       1 cells - ;

: mem-save ( x1 ... xn n -- addr )
   dup 1+ cells allocate Abort" no more memory"  \ x1 ... xn n addr
   dup >r   2dup ! cell+               \ store length
   swap 0 ?DO  swap over ! cell+  LOOP  ( addr1 )
   drop r>
;
: mem-restore ( addr -- x1 ... xn n )
   dup >r dup @                     ( addr len )
   dup >r cells +                   ( addr' )
   r@ 0 ?DO  dup @ swap cell-  LOOP  ( x1 ... xn addr )
   drop r>                          ( x1 ... xn n )
   r> free Abort" mem-restore: free memory failed"
;
\ ------------------------------------------------------------------------
\ Compilation on demand on gforth 0.6.2
\ ------------------------------------------------------------------------

Variable Libfile    \ pointer to library filepath string ( libpath )
Variable Libgroup   \ points to the following data structure:
                    \ | ptr->libpath | dposition | current | search-order |
Variable Libload    \ true when lib-loading
Variable Libdepth   \ Checking stack integrity while pre-loading a library

\ ------------------------------------------------------------------------
\ gforth_0.6.2 internals used:

\ Variable Loadfile            \ fid of the source file being included
\ Dvariable Loadfilename       \ points to <path/filenam> of loadfile
\ Variable #fill-bytes         \ # of chars placed at source by the last refill
\ : push-file  ( -- )          \ push source context on return stack
\ : pop-file   ( n -- n' )     \ pop source context from return stack
\ : read-loop  ( i*x -- j*x )  \ refill and interpret a source file until eof
\ Defer Parser ( caddr u -- )  \ Parser is the interpreter/compiler used by included
\
\ ------------------------------------------------------------------------
\ input stream and dictionary manipulations
\ ------------------------------------------------------------------------

: source-position  ( -- udpos )  \ gforth: Loadfile #fill-bytes
   source-id 0= IF  >in @ 0  EXIT THEN
   Loadfile @ file-position throw   #fill-bytes @ >in @ - 0 d-
;
: libgroup,  ( -- )  \ build data structure to remember source code and context
   Libfile @ ,   source-position , ,
   current @ ,   get-order dup 1+ 0 DO , LOOP
;

\ group_addr: | ptr->libpath | dposition | current | search-order |

: >libgroup   ( group_addr -- )  \ gforth: Loadfile Loadfilename
   dup @ count   2dup R/O open-file throw Loadfile !   Loadfilename 2!
   cell+ dup 2@ Loadfile @ reposition-file throw
```

```
118     cell+ cell+ dup @ current !
119     cell+ dup @ dup >r cells + r> 1+ 0 DO  dup @ swap cell-  LOOP drop   set-order
120   ;
121   : load-libgroup  ( i*x group -- i*x )  \ gforth: push-file Loadfile read-loop pop-file
122     Libload @ >r   Libload on
123     get-order current @ swap 1+ mem-save >r
124     push-file   >libgroup ['] read-loop catch
125                 Loadfile @ close-file throw
126     pop-file throw
127     r> mem-restore 1- swap current !   set-order
128     r> Libload !
129   ;
130   Create Tmarker  ( here ) 0 , ( depth ) 0 ,  ( input stream ) 6 cells allot
131
132   : mark-target  ( -- )
133     here Tmarker !   depth Tmarker cell+ !
134     -2 >in +!  save-input  2 >in +!  \ back up to re-interpret : after restore-target
135     Tmarker cell+ cell+  over 1+ 0 DO  tuck ! cell+  LOOP  drop
136   ;
137   \ Tmarker: | target-dp | depth | save/restore-input arguments |
138
139   : restore-target  ( i*x -- i*x )
140     postpone [   Tmarker @ Dp !                                    \ restore here
141     Tmarker cell+ @ >r BEGIN  depth r@ - WHILE  drop  REPEAT  rdrop   \ restore depth
142     Tmarker cell+ cell+ dup @ dup cells rot +
143     swap 1+ 0 DO  dup @ swap cell-  LOOP  drop  restore-input throw   \ restore input stream
144   ;
145   \ ------------------------------------------------------------------------
146   \ Defining Library words and Predefinitions
147   \ ------------------------------------------------------------------------
148
149   wordlist Constant Predefined  \ only these words get searched during lib-loading
150
151   :noname ( -- )  ;
152   Constant #:    \ Predefinition type, always executes a noop
153
154   Variable /dev0
155   :noname ( -- addr )  /dev0 ;
156   Constant #var  \ Predefinition type, returns a memory cell address
157
158   : Libpredef  ( <name> type -- )
159     get-current >r   Predefined set-current   Create ,   r> set-current
160   Does> ( i*x -- j*x )  @ execute ;
161
162   : Libdef  ( <name> -- )
163     Create immediate   Libgroup @ ,
164   Does> ( -- )
165     state @ 0= abort" Libdef can not be interpreted"
166     @ >r restore-target r> load-libgroup
167   ;
168   : Libcreate  ( <name> type -- )
169     >in @ >r   Libdef
170   \ cr ." Libcreate "  r@ >in !  parse-name type
171     r> >in !   Libpredef
172   ;
173   \ ------------------------------------------------------------------------
174   \ Library loading
175   \ ------------------------------------------------------------------------
176
177   : pre-compile ( addr u -- )
178     [struct]-voc lookup @ = IF  evaluate  EXIT THEN  \ gforth bug fix for [IF] etc.
179     2dup Predefined search-wordlist IF  nip nip execute  EXIT THEN
180     integer? drop                                  \ skip all undefined words but leave numbers
181   \   2dup 2>r  integer? 0= IF  2r> type ."  ?" true throw THEN  rdrop rdrop
182   ;
183   : library  ( <name> -- )   \ gforth: parser
184     depth 1+ Libdepth !
185     parse-name dup 0= abort" file name required"
186     here   dup Libfile !   over 1+ allot   place    \ store path/file name
187     ['] pre-compile IS parser   Libfile @ count ['] included catch
188     postpone [   dup IF  cr ." libload error"  THEN  throw
189   ;
190   : :   ( <name> -- )  mark-target : ;
191
192   : ~  ( -- )   \ skip the rest of the input file when pre-compiling a library
193     Libload @ IF  BEGIN  refill 0= UNTIL  THEN
```

```
194  ;
195  Create ";  ( -- saddr )  ," ;"
196
197  : scan-for-;  ( -- )
198     BEGIN  BEGIN  BL word dup c@ 0= WHILE  drop refill 0= throw  REPEAT
199            count "; count compare 0=
200     UNTIL
201  ;
202  \ ------------------------------------------------------------------------
203  \ Only Predefinitions are executed during initial library-loading
204  \ ------------------------------------------------------------------------
205  Predefined set-current
206
207  ' only        Alias only
208  ' definitions Alias definitions
209  ' also        Alias also
210  ' Forth       Alias Forth
211  ' Root        Alias Root
212  ' library     Alias library
213  ' library     Alias include
214  ' decimal     Alias decimal
215  ' hex         Alias hex
216  ' (           Alias (
217  ' \           Alias \
218  ' .(          Alias .(
219  ' cr          Alias cr
220  ' 2drop       Alias !
221  ' noop        Alias @
222  ' drop        Alias ,
223  ' drop        Alias allot
224  ' [IF]        Alias [IF]
225  ' [ELSE]      Alias [ELSE]
226  ' [THEN]      Alias [THEN]
227  ' noop        Alias immediate
228
229  : ~   ( -- )
230     Libdepth @ depth - abort" stack imbalance during precompilation"
231     here Libgroup ! libgroup,
232  ;
233  : Constant  ( n -- )
234     get-current >r   Predefined set-current   Constant   r> set-current
235  ;
236  : Variable  ( -- )     #var Libcreate ;
237
238  : Create    ( -- )     #var Libcreate ;
239
240  : :         ( -- )     #: Libcreate scan-for-; ;
241
242  \ ------------------------------------------------------------------------
243  \ debugging words
244  \ ------------------------------------------------------------------------
245  Forth definitions
246
247  : clear   ( xxx -- ) BEGIN  depth WHILE drop REPEAT ;
248  : .lib    ( -- )     Libfile @ ?dup IF  count type  THEN ;
249  : .group  ( -- )  \ group: | ptr->libpath | dposition | current | search-order |
250     Libgroup @ ?dup 0= ?EXIT
251     dup @ count type   dup 4 cells + @ 5 + cells dump
252  ;
253  : predefs ( -- )     Predefined wordlist-words ;
254
```

lib.fs

```
1  cr .( pre-compiling lib.fs )
2
3  ~ : dudel  ." didel " .( 1 ) daddel @ . .( 2 ) ;
4
5  ~ 2 Constant Konstant   Variable daddel  Konstant daddel !
6
7  ~ Variable resource   2 resource !
8
9  ~ : rumpel  ( -- )  ." alle " resource @ . ;
```

test.fs

```
1   \ ************************************************************
2   \ Testing the compilation on demand mechanism
3   \ ************************************************************
4   Only Forth also definitions
5
6   [IFDEF] empty empty [ENDIF] Marker empty
7
8   include library.fs
9
10  Vocabulary lib  lib definitions
11
12  library lib.fs
13
14  Forth definitions lib also
15
16  : drebel   [ cr ] .( a ) dudel ." hei " ;
17
18  : daudel   ." ta" dudel ." dei " daddel .( b ) @ . ;
19
20  : doedel   .( c ) rumpel  .( d )  1 resource +! ;
21
22  cr .( finished compiling, words in lib Vocabulary: ) words
23
24  cr drebel cr daudel cr doedel cr doedel
25
26  Previous
```

Tags

Matthias Trute

Wir (ich entsinne mich an Bernd und Erich) hatten eine unserer mehr oder weniger regelmäßigen Chatsessions, wo es hinreichend oft um Forth–Themen geht. Aber nicht nur. So haben wir irgendwann einmal darüber philosophiert, was Tags eigentlich sind. Also die Dinger, die in Twitter und ähnlichen Medien gerne und oft benutzt werden. Die unvermeidliche Frage kam dann auf, was hat das mit Forth zu tun? Und so haben wir rein spekulativ und ohne jeglichen Codebezug Tags in Forth eingeführt. Herausgekommen ist eine kleine und eher spröde Wiki-Seite ***http://wiki.forth-ev.de/doku.php/words:tags*** *, die es lohnt aufbereitet zu werden.*

Was sind Tags?

Tags, oft auch Hashtags genannt, sind Attribute, die durch ihre pure Existenz eine Information transportieren. Häufig werden sie durch ein Sonderzeichen wie `@` oder # (daher der Name: # heisst Hash) eingeleitet, damit sie einfach erkennbar sind und vor allem keine Verwechslung mit normalen Worten verursachen. Je nach Umfeld werden aus diesen Tags dann Links, die zu anderen Texten mit dem gleichen Tag führen. Bei Forth gibt es diverse Flags wie das Immediate, die genauso eine Zusatzinformation transportieren. Chuck Moore hat die Idee mit den Farben in Colorforth eingeführt. Rote Worte verhalten sich anders als grüne.

Flags und Farben sind mit Tags vergleichbar. Sind sie vorhanden, haben die betreffenden Worte bestimmte Eigenschaften, die sie sonst nicht haben. Das alles sind wohlgemerkt zusätzliche Informationen. Der betreffende Code wird davon nicht beeinflusst. Als mögliche Einsatzfelder für Tags haben wir ad hoc die folgenden gefunden:

- Immediate Flag (und andere)
- Optimiererhinweise (Konstantenfaltung etc)
- Wortlistenmitgliedschaft
- Farben (die von Colorforth)
- OO Varianten

Glossar

Das Ganze ist natürlich nicht sonderlich ausgereift. Damit man aber einen Einstieg hat, haben wir ein einfaches Glossar definiert und ein paar Eckdaten festgelegt.

Tags sind an Wortdefinitionen gekoppelt. Das heißt, nur ein Wort mit einem Namen kann ein oder mehrere Tags haben. Sie selbst bestehen aus einer ID, die einen Namen haben kann. Diese ID wird als Attribut den Wortdefinitionen als Liste beigeordnet. Als Aufhänger haben wir das Name Token von Forth 2012 genommen. Das hat den kompletten Zugang zu allen Headern eines Wortes. Die schon lange vorhandenen Execution Tokens sind hingegen zu low level, da muss man eher raten, ob es die gewünschten Informationen überhaupt gibt.

Die `tag-id` ist eine zellige Angabe ohne nähere Bewertung des numerischen Werts. `nt` ist das Name Token.

```
create-tag ( -- tag-id )
```

legt ein neues Tag an. Analog zu wordlist. Kann später einen Namen via `constant` erhalten.

```
assign-tag ( nt tag-id -- )
```

weist ein Tag einem Wort zu. `nt` ist dessen Name Token.

```
remove-tag ( nt tag-id -- )
```

entfernt ein Tag von einem Wort.

```
delete-tag ( tag-id -- )
```

Entfernt ein Tag komplett. Etwas problematisch, auch wordlists leben ewig.

```
map-tag ( i*x tag-id xt -- j*y f )
```

wobei `xt` den Effekt `(i*x nt -- j*y flag)` hat. Ein allgemeiner Iterator. Für alle Worte mit dem Tag `tag-id` mach was, `flag` ermöglicht das vorzeitige Verlassen der Schleife. Siehe `traverse-wordlist`. ist flag true (−1) wird weiter iteriert, ist es false (0) wird die Iteration vorzeitig beendet. `f` ist die Information, ob es einen solchen vorzeitigen Ausstieg gab (−1) oder nicht (0).

```
has-tag? ( nt tag-id -- f )
```

prüft, ob ein Wort mit dem Name Token NT mit einem (bekannten) Tag markiert ist.

```
get-tags ( nt -- tag-n tag-1 n )
```

Lies alle Tags eines Wortes `nt` aus.

```
set-tags ( tag-n tag-1 n nt -- )
```

Überschreibe alle Tags eines Wortes `nt`.

Beispiele

Immediate wird zum (Hash–) Tag `#immediate`. Mittels `#immediate has-tag?` kann der Interpreter diese Eigenschaft abfragen und darauf reagieren. Das #–Zeichen ist dabei pure Konvention, es wird nicht als solches gebraucht.

```
create-tag constant #immediate
s" IF" find-name ( -- NT) #immediate assign-tag

: INTERPRET ...
  STATE @ IF ( NT -- )
    DUP #immediate has-tag? IF
      NAME>INTERPRET
    ELSE
      NAME>COMPILE
    THEN
    EXECUTE
....
;
```

Alle Worte mit dem Tag `FOO` sind in der Wortliste `FOO`. Damit kann ein Wort in mehreren Wortlisten vertreten sein, ohne dass man zu Aliasen oder Definitionen wie

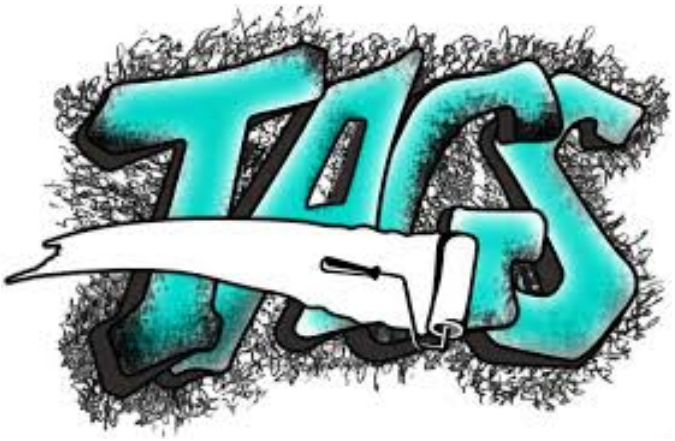
```
: dup dup ;
```

Zuflucht nehmen muss. Die Wortliste aller immediate Worte gibt es automatisch dazu. Und die Worte verhalten sich auch immer gleich. Ein Immediate Wort bleibt es, egal in welcher Wordlist es registriert ist.

```
: show-words-with-tag ( tag-id -- )
  [: name>string type -1 ;] map-tag DROP
;

\ display all immediate words
#immediate show-words-with-tag
```

Ausblick

Derzeit gibt es kein Forth, das Tags nutzt oder bereitstellt. Lohnt es sich, darüber nachzudenken? Was sollte man anders machen? Was geht einfacher?



Forth–Gesellschaft e.V. Ordentliche Mitgliederversammlung 23.04.2017

Erich Wälde

Moderation Martin Bitter

Protokollant Erich Wälde

Teilnehmer
Direktorium: Ewald Rieger, Ulli Hoffmann, Bernd Paysan
insgesamt 14 stimmberechtigte Mitglieder (Anzahl der ausgegebenen Stimmkarten)

Sitzungsdatum 23.04.2017

Sitzungsbeginn 09:00 h

Sitzungsende 11:40 h

Sitzungsort Kernwasser Wunderland Freizeitpark GmbH
Griether Str. 110-120 D 47546 Kalkar

Begrüßung

Ewald Rieger begrüßt im Namen des Direktoriums die anwesenden Mitglieder.

Wahl des Schriftführers

Als Protokollant wird Erich Wälde gewählt.

Wahl des Versammlungsleiters

Zum Sitzungsleiter wird Martin Bitter gewählt. Der Sitzungsleiter stellt fest, dass die Versammlung fristgerecht einberufen wurde. Es wurden 14 von 100 Stimmkarten ausgegeben. Damit sind mehr als 10 % der Mitglieder anwesend und die Versammlung ist beschlussfähig.

Ergänzungen zur Tagesordnung

Anträge zur Ergänzung der Tagesordnung liegen keine vor, die Tagesordnung wird einstimmig angenommen.

Bericht des Direktoriums

a. Bericht der Verwaltung (Ewald Rieger)

Satzungsänderung Die in Augsburg beschlossene Umformulierung des Vereinszwecks wurde am 23.3.2017 beim zuständigen Registergericht (Hamburg) eingetragen.

Mitgliederentwicklung Im Jahr 2016 gab es immerhin einen Neuzugang und 5 Austritte (davon einer durch Ableben des Mitglieds, zwei weitere aus gesundheitlichen Gründen). Der Verein hatte zum Jahresende 2016 100 Mitglieder. In 2017 sind bis dato 1 Austritt und 1 Zugang zu verzeichnen. Die Mitgliederzahlen weisen weiterhin nach unten (27 Mitglieder weniger seit 2007). Es ist unklar, ob dieser Trend überhaupt aufgehalten werden kann.

Finanzen Ewald Rieger erläuterte im Detail die Einnahmen und Ausgaben, getrennt nach Verein und Zweckbetrieb (Vierte Dimension). Es ergibt sich ein Vermögen des Vereins von 6719.75 € zum Jahresende. Bei der letzten Versammlung wurde angesagt, dass alle Ausgaben der Zustimmung durch Ewald Rieger bedürfen. Das hat sich als so wirksam erwiesen, dass die für die *Forth–Bildung* vorgesehenen 2000 € nicht angetastet wurden (insbesondere wurde keine Erstattung von Reisemitteln beantragt).

Für das Jahr 2017 sind Einnahmen von etwa 4390 € und Ausgaben in Höhe von 6620 € zu erwarten. In den Einnahmen wird die 2016 beschlossene Erhöhung der Beiträge erstmals sichtbar. Damit ist die Finanzierung der Vierten Dimension (inkl. Versand) wieder abgedeckt.

b. Rund um das Forth Magazin (Ulrich Hoffmann)

Die *Vierte Dimension* erscheint weiterhin, sie ist weiterhin die letzte gedruckte Veröffentlichung in Sachen Forth auf dem Planeten. Dank geht ausdrücklich an Michael Kalus, der vor allem Inhalte für das Heft beschafft und sich dann auch um die Herstellung kümmert. Es hat sich gelohnt, die Produktion des Heftes mit LaTeX und verwandten Werkzeugen hochgradig zu automatisieren.

Die Zusammenarbeit mit James Bowman von der *Silicon Valley FIG* hat leider keine englischsprachigen Hefte erzeugt, vermutlich fehlen da Personen, die ihre Zeit freiwillig einbringen. Es ist andererseits zu sehen, dass der Anteil an englisch–sprachigen Artikeln in der Vierten Dimension steigt. Die Hefte erfahren jeweils ca. 500 Downloads von der Webseite.

Die Bereitstellung des Heftes in einem epub–Format hat sich als eher schwierig herausgestellt. Wenn das jemand vorantreiben will, nur zu.

Es wird wie jedes Jahr ausdrücklich gewünscht, dass die Vorträge zur Tagung in Textform für die Vierte Dimension eingereicht werden.

c. Internet–Präsenz (Ulrich Hoffmann)

Gerald Wodni hat eine neue Webseite erstellt, die ist derzeit unter `http://neu.forth-ev.de` erreichbar. Es sind noch nicht alle Inhalte migriert. Es wird ausdrücklich

gewünscht, die neue Seite zu verwenden und Fehler/Unzulänglichkeiten an Ulli Hoffmann zu berichten. Die neue Seite bietet ein paar neue Möglichkeiten, die genaue stilistische Ausgestaltung ist noch zu tun.

Damit sind wir der Umstellung der Webseite von `geeklog` weg ein deutliches Stück näher gekommen. Die Umschaltung der Webseite und die Abschaltung des alten Servers ist für diesen Sommer vorgesehen. Der Maillisten–Server kann jederzeit vom alten auf den neuen Server umgezogen werden (Bernd Paysan), das wiki und das repository sind schon umgezogen.

Erwähnenswert ist in dem Zusammenhang auch, dass die Webseiten von Charles Moore und Chen Ting inzwischen offline sind. Um so wichtiger ist das Sammeln von Inhalten und deren öffentlicher Zugang.

d. Außendarstellung und Projekte (Bernd Paysan)

Die Forth Gesellschaft war 2016 an mehreren Veranstaltungen präsent:

Die Forth–Tagung 2016 fand zeitgleich mit der Retropulsiv und dem Linuxtag in Augsburg statt. Leider hat sich durch dieses Zusammentreffen nicht das erhoffte, zusätzliche Publikum eingefunden. Auch an dem von Martin Bitter angebotenen workshop fand sich lediglich ein Interessent ein.

Das 17. Vintage Computing Festival Europe fand am 30.4.2016 in München statt, und zwar zum letzen Mal am gewohnten Ort (inklusive der ausgesprochen retro–mäßigen sanitären Einrichtungen). Die Mehrzweckhalle des Eisenbahner-Sportclubs München-Ost wurde inzwischen abgerissen.

Die Maker Faire (Hannover, 3 Tage) hatte den Freitag ausdrücklich als Schüler–Tag ausgerichtet. Dementsprechend waren viele Schulklassen unterwegs. Ausgestellt wurde der Triceps sowie eines der Experimente zum Algenwachstum (von Matthias Koch). Bernd Paysan schätzt, dass Neumitglieder am ehesten aus den Kreisen der *Maker* gewonnen werden können. Die Erstellung/-Überarbeitung eines Faltblatts wurde angeregt.

33C3 (Hamburg, 4 Tage, incl. Bitkanone und Beteiligung von net2o/Bernd Paysan am Netzwerk Track `#wefixthenet`). Es war im Vorfeld sehr schwierig, überhaupt Karten zu erhalten. Dieses mal waren wir in einer ganz anderen (und finsteren!) Ecke untergebracht und hatten dadurch überhaupt kein Lauf–Publikum. Das ist einerseits sehr schade, wurde andererseits für diverse andere Aktivitäten genutzt. Unsere Tischnachbarn von der Hochschule Rhein–Main wurden von ihrem Professor explizit mit Forth konfrontiert. Es ist derzeit noch unklar, wo der 34C3 stattfinden wird. Klar ist nur, dass das Gebäude in Hamburg inzwischen abgebrochen wurde. Das neue Gebäude soll leider nicht mehr über die großen Hörsäle verfügen, es ist damit wahrscheinlich nicht mehr für diese Veranstaltung geeignet.

Für 2017 ist die Beteiligung am VCFe (29.4.–1.5.2017), an der Maker Faire in Hannover (25.–27.8.2017) sowie auf dem 34C3 (27.–30.12.2017) geplant. Ein workshop an der Hochschule Augsburg könnte ebenfalls wieder stattfinden

Es gibt derzeit keine offiziell geförderten Projekte.

e. Bericht des Kassenprüfers (Klaus Zobawa)

Die Prüfung der Kasse fand am 22.04.2016 durch Klaus Zobawa statt. Er berichtet, dass alle Buchungen belegt und nachvollziehbar sind. Die Buchhaltung ist vorbildlich.

Entlastung des Direktoriums

Klaus Zobawa stellt den Antrag, das Direktorium zu entlasten.

Der Antrag wird einstimmig angenommen, das Direktorium ist damit entlastet.

An dieser Stelle wurde die Sitzung unterbrochen. Der Tradition folgend hatte der Drachenrat getagt. Thomas Prinz übergab den Drachen an den neuen Drachenhüter Klaus Zobawa. Die erste Amtshandlung des neuen Drachenhüters war die ebenso traditionelle Aufstellung zum Gruppenfoto, welche unmittelbar nach der Sitzung stattgefunden hat.

Wahl des Direktoriums

Die bislang amtierenden Mitglieder des Direktoriums sind bereit, für eine weitere Amtszeit zu kandidieren.

Das Direktorium (Ewald Rieger, Ulli Hoffmann, Bernd Paysan) wird einstimmig für eine weitere Amtszeit gewählt. Alle Gewählten nehmen die Wahl an.

Projekte

Die **Maker Faire 2017** in Hannover findet am 25.–27.8.2017 im Congress Centrum (HCC) statt. Der Freitag ist explizit für Schülerbesuche vorgesehen. Wer mitmachen will (Standbesatzung) meldet sich bei Ulrich Hoffmann.

Der Jahreskongress des Chaos Computer Clubs **34C3** findet vom 27.–30.12.2017 irgendwo im erreichbaren Teil der Galaxis statt (Nachtrag: in Leipzig).

Gerald Wodni berichtet über den von ihm entwickelten und am 33C3 vorgestellten **BitBot**. Das ist ein kleiner, fahrender Roboter mit einem 5×5 RGB–LED–Feld auf der Oberseite. Auf diesem kann man z.B. ein Labyrinth aufmalen, durch das der Roboter gelenkt werden muss. Er möchte dieses Ding als Bausatz realisieren (unter 40 €). Der kann als Schulungs–Platform zum Einsatz kommen, inklusive Zusammenlöten, bevorzugt an Stellen, an denen der Verein vertreten ist. Er soll zum Selbstkostenpreis abgegeben werden. Verschiedene Zusagen wurden gemacht (Mithilfe bei Platinen–Layout, Frästeilen). Der Verein wird die Beschaffung von einigen Platinen unterstützen, die Kosten dafür sollten moderat ausfallen. Das ganze soll kein Produkt werden — das würde der Gemeinnützigkeit widersprechen.

Ein weiteres Projekt wäre die Entwicklung von **Mini-Bitkanonen**, also etwa 8×8 o.ä. große RGB–LED–Felder, die jeweils autonom mit einem Kontroller ausgestattet sind. Sie können also einzeln verwendet werden. Allerdings ist die Idee durchaus, mehrere davon zu einer Kette zu kombinieren und ein *gemeinsames* Ergebnis herzustellen (eine Buchstaben– oder Ziffernanzeige, ein snake–Spiel mit kombiniertem grösserem Feld, eine Farbenlampe etc). Dieses Projekt wäre möglicherweise geeignet, um eine kleine Referenz–Implementierung für ein Netz von verteilten Knoten zu realisieren — nach dem Vorbild von Open Forth Net oder dem von der Universität Palermo vorgestellten *wireless sensor network* (VD 2016-01).

Der **Wirtschaftsplan** kam dieses Mal etwas zu kurz. Es wird von uns erwartet, dass wir die zu Verfügung stehenden Mittel *zeitnah* im Sinne des Vereinsanliegens ausgeben. Daher ist die Planung der Ausgaben immens wichtig. Wer also Ausgaben in diesem Sinne vorhat oder vorschlägt, sollte sich zügig an das Direktorium wenden.

Verschiedenes

Die **Tagung 2018** könnte möglicherweise im Linux–Hotel in Essen stattfinden (Carsten Strotmann fragt nach möglichen Terminen).

Alternativ, aber lieber erst im Jahr 2019 gäbe es einen Veranstaltungsort in Worms (Ewald Rieger).

Die **EuroForth 2017** findet vom 8. bis zum 10. September 2017 in Bad Vöslau (Östereich) statt.

Carsten Strotmann stellt einen microblogging Service vor (Alternative zu Twitter). Er beruht auf GNU social, verwendet das OStatus Protokoll und hat mit `mastodon` ein Bedien–GUI bekommen. Dieses hat sich als so gut herausgestellt, dass `mastodon` die derzeit am schnellsten wachsende online–community ist. Die Serverseite ist auf viele Instanzen aufgeteilt, jeder kann seinen eigenen Server betreiben, wenn er will. Wer einen account will oder Fragen hat, meldet sich bei Carsten.

Erich Wälde wird die Edition der VD 2017-03 übernehmen.

Es wurde angeregt, eine weitere Bestellung von T–Shirts aufzulegen. Zum einem mit dem Drachen von Bernd Paysan, zum anderen auch mit dem neuen Logo von `theforth.net` (Gerald Wodni), welches auch von der Euroforth schon genutzt wird. Bernd Paysan wird eine begrenzte Auswahl an Ausführungen (nicht jeder sein eigenes Material) zusammenstellen und die Bestellung anleiern.

Schluss

Die Jahresversammlung endet um 11:40 h.

Hechingen, den 14.05.2017

gez. Erich Wälde

Abbildung 1: Gruppenfoto vor dem Kühlturm

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) – 41 15 46 53
`bernd.paysan@gmx.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00 in der Pizzeria La Capannina, Weitlstr. 142, 80995 München (Feldmochinger Anger).

Hamburg **Ulrich Hoffmann**
Tel.: (04103) – 80 48 41
`uho@forth-ev.de`
Treffen alle 1–2 Monate in loser Folge
Termine unter: `http://forth-ev.de`

Ruhrgebiet **Carsten Strotmann**
`ruhrpott-forth@strotmann.de`
Treffen alle 1–2 Monate Freitags im Unperfekthaus Essen
`http://unperfekthaus.de`.
Termine unter: `http://forth-ev.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

Forth-Hardware in VHDL microcore (uCore)	**Klaus Schleisiek** Tel.: **(0 75 45) – 94 97 59 3**$_p$ kschleisiek@freenet.de
KI, Object Oriented Forth, Sicherheitskritische Systeme	**Ulrich Hoffmann** Tel.: **(0 41 03) – 80 48 41** `uho@forth-ev.de`
Forth–Vertrieb volksFORTH ultraFORTH RTX / FG / Super8 KK–FORTH	Ingenieurbüro **Klaus Kohl–Schöpe** Tel.: **(0 82 66)–36 09 862**$_p$ mcForth

Termine

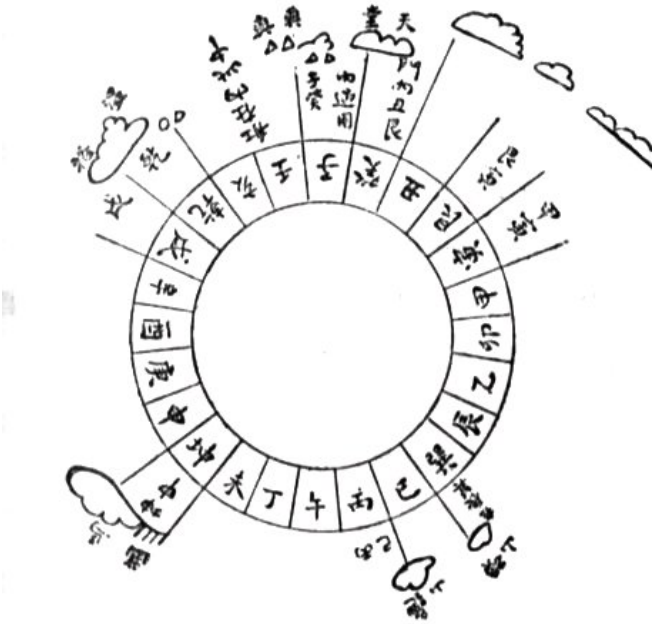
Donnerstags ab 20:00 Uhr
Forth–Chat net2o `n2o chat forth@bernd` mit dem Key `n2o keysearch kQusJ`,
voller Key: `kQusJzA;7*?t=uy@X}1GWr!+0qqp_Cn176t4(dQ*`

25.–27. August 2017
Maker Faire Hannover
`http://www.makerfairehannover.com`

08.–10. September 2017
EuroForth
`http://euro.theforth.net/`

27.–30. Dezember 2017
34c3 Chaos Communication Congress in Leipzig
`https://events.ccc.de/tag/34c3`

Details zu den Terminen unter `http://forth-ev.de`



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.

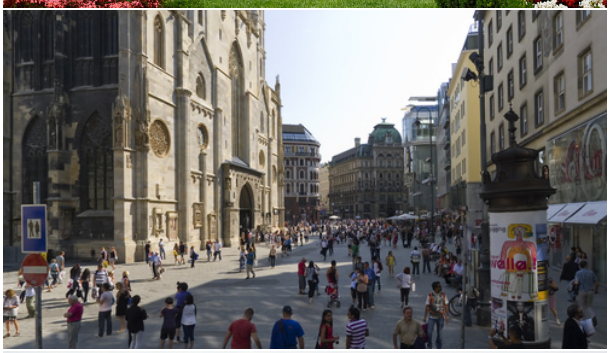
Einladung zur

EuroForth 2017 vom 08. bis 10. September in Bad Vöslau, Österreich

The 33rd EuroForth conference takes place in the spa town Bad Vöslau (near Vienna) in Austria.

`http://euro.theforth.net/`

The conference will be preceeded by the Forth standards meeting which starts on the 6th of September. Both the meeting and the conference will be hosted in the College Garden Hotel. The exact address for your navigation system is: Johann Strauß–Straße 2, 2540 Bad Vöslau, Austria.



Programme

Forth standard meeting

Wednesday, 6th September 12:00 –
Friday, 8th September 12:00

EuroForth conference

Friday, 8th September 13:00 –
Sunday, 10th September 14:00

Saturday Vienna excursion

Besides having wonderful discussions with other splendid Forthers, we will roam Vienna intensively saturday afternoon and evening: Starting with a guided tour through Schloss Schönbrunn (Austria's No. 1 attraction), then walk through Vienna's city center and will conclude in the Wiener Prater.

Bring your partners!

This years conference will feature a complete separate track for a hacker's partner, including but not limited to: Enjoying a great view over Vienna from the Donauturm, Shopping at the famous Mariahilferstraße, Treasure chambers, Tiergarten Schönbrunn (we got pandas!), before we meet the group for a guided tour of the palace itself!

Images:

1. Therme Bad Vöslau: Wolfgang Glock / CC BY 3.0
2. Château de Schönbrunn, Vienne (Autriche). Vue depuis les jardins: Yelkrokoyade / CC BY–SA 3.0
3. Wiener Stephansplatz: Gugerell / CC0 1.0
4. Tiergarten Schönbrunn: Manfred Werner / GNU FDL