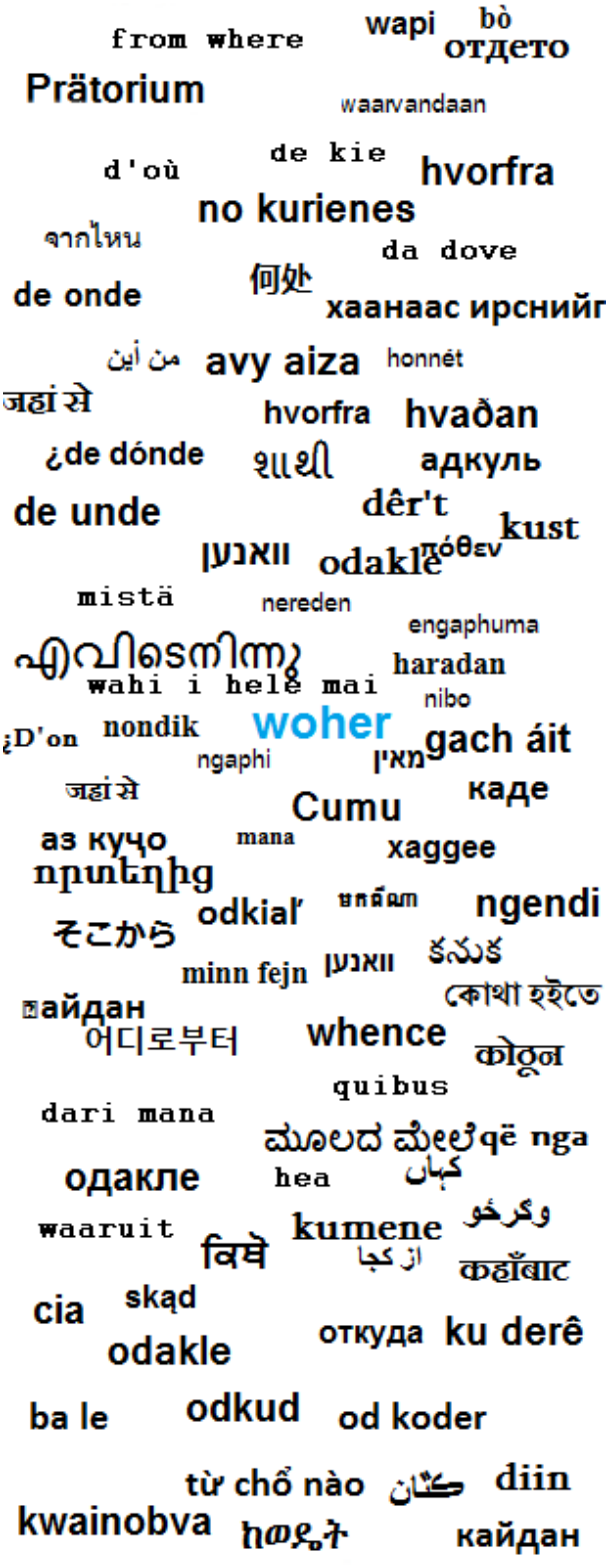
Vierte Dimension

2/2018

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 M. Kalus

Quelle: Vom Autor zu beziehen.

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,

50 Jahre Forth — ein langer Weg. Woher sind wir gekommen? Was ist der Stand der Dinge? In Schottland wird das dieses Jahr genauer betrachtet werden, vor allem, wenn Charles Moore wirklich, wie geplant, dort hinkommen sollte. Hoffentlich können viele von Euch auch dort sein bei den Gesprächen am Kamin auf der Euroforth in Edinburgh im September.

Wer das alles genauer nachlesen möchte, wird bei Amazon fündig. Jürgen Pintaske hat in jüngster Zeit viele der vom Untergang bedrohten Arbeiten über Forth als eBooks herausgegeben. Auch das Buch von Charles Moore selbst: „The Early Years“.

Dass Forth die Stelle eines Betriebssystems einnimmt, war ja am Anfang so. Als es noch keine GUIs gab, und die Rechner, ohne Fenster, einfach so auf dem Bildschirm ihre Zeilen gemalt haben. Auf der Euroforth in Edinburgh, im September, kann man nachspüren, wie es sich angefühlt haben könnte. DOCKER bietet die Möglichkeit, Forth im Container auf dem Rechner zu haben. Matthias Trute lässt uns teilhaben daran, wie man das eingerichtet bekommt.

Der Wunsch nach grafischen Darstellungen war damals natürlich auch da, wurde aber zunächst anders gelöst als mit „graphischen“ Oberflächen. Dass das mit den Pixeln immer noch geht, weils in den Terminals steckt, zeigt Carsten Strotmann.

Und Fred Behringer schließt sich gleich an und untersucht die Zeichengestaltung, die man mit einem Terminal machen kann, das die ganze Palette der ESC–Steuerzeichen zur Verfügung hat.

Für Nachwuchs — auch für Forth — sorgte Martin Bitter. Enkeln spielerisch Forth nahe zu bringen, ist ein guter Versuch, finde ich. Möge es gelingen, über die Puppenküche hinaus das Interesse an Forth auch zukünftig aufrecht zu halten. Wir werden weiter darüber berichten ... :-)

Wie es mit Forth weitergehen wird, das steht in den Sternen. Bin gespannt aufs nächste Heft, in dem dann hoffentlich schon Neues aus Schottland berichtet werden kann.

Auf unserer Jahrestagung, der Forth–Tagung in Essen, hat Ewald Rieger bekannt gegeben, dass er von der Vorstandsarbeit zurücktritt. Das ist bedauerlich. Er geht gesund in den Ruhestand. Das ist erfreulich. Also sei es ihm gegönnt. Im Protokoll ist die Wahl zum neuen Vorstand nachzulesen: Neu gewählt wurde Carsten Strotmann, Bernd Paysan und Ulrich Hoffmann wurden wiedergewählt.

Und unser bronzener Swap–Drachen, Bewacher der Schatulle, ging zu Andrea Rieger. Wie sich das zugetragen haben könnte, darüber gehen zwar Gerüchte, aber nichts Genaues weiß man nicht. Bis heute ist noch nie etwas Verlässliches diesbezüglich aus dem Drachenrat nach außen gedrungen. Wie dem auch sei: Herzlich willkommen in der Runde, liebe Andrea. Bleib uns gewogen.

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-template

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

Ulrich Hoffmann
Bernd Paysan
Carsten Strotmann

Kontakt: Direktorium@Forth-ev.de

Tagung des Forth e.V. — Erfahrungen und Gedanken eines Frischlings

„Ich bin spät dran“, dachte ich, als ich die Anmeldung zur Forth–Tagung ausfüllte. Dieses Jahr fand das jährliche Treffen in der Stadt Essen, also mitten im Ruhrpott, statt. Eigentlich hatte ich geplant, zuhause zu übernachten und jeden Tag die 120 Minuten für Hin- und Rückfahrt in Kauf zu nehmen, um mir das Hotel zu ersparen. Ich mag Hotels nicht besonders und brauche immer ein paar Tage, um mich zu akklimatisieren. Aber es kam anders. Jemand vom Verein kontaktierte mich und fragte, ob er sich um ein Zimmer für mich kümmern solle. Ich nahm das Angebot an. Das war die beste Entscheidung überhaupt. Die Unterbrechungen und der Stress durch die zusätzlichen Fahrten hätten ein intensives Erleben der Tagung unmöglich gemacht.

In der Nacht vor der Tagung habe ich unruhig geschlafen. Wie wird es sein? Wird sich jemand meiner annehmen oder werde ich links liegen gelassen? Werden mich die Leute spüren lassen, dass ich nur elementare Kenntnisse der Sprache Forth habe oder mich als gleichwertigen Gesprächspartner akzeptieren? Ist es das alles überhaupt wert oder hätte ich lieber so wie bisher alleine in meinem stillen Kämmerlein herumprutscheln sollen?

Auf der Autofahrt nach Essen verflogen dann die Sorgen und Bedenken und die Vorfreude auf das Kennenlernen der Forth–Gesellschaft bahnte sich ihren Weg an die Oberfläche. Ich meine mich daran zu erinnern, dass ich ein breites Grinsen im Gesicht hatte.

Bei der Ankunft am Linux–Hotel Essen dann der Wow!–Effekt. Das Hotel ist in einer alten Villa mit Nebengebäude untergebracht. Ein privater Park lädt zum Verweilen ein. Ein Traum. Der Betreiber hat sich voll GNU/Linux und dem Open–Source–Gedanken verschrieben. Forth ist seit jeher Open–Source. Passt!

Aber ich hatte noch etwas zu erledigen. Persönlicher Erstkontakt mit dem Verein. Den Ersten, welchen ich sah, grüßte ich sofort mit seinem Vornamen. Ich kannte ihn von einem Vortragsvideo auf der Internetseite des Forth e.V. Sein Gesichtsausdruck war Gold wert. Es ist dieser typische Ausdruck, wenn man von jemanden angesprochen wird und sich beim besten Willen nicht erinnern kann, aber meint, man müsste es wissen. Ich habe die Sache dann aufgeklärt. Der Rest war ein Kinderspiel. Das sage gerade ich, der sonst eher Schwierigkeiten hat, auf Menschen zuzugehen. Die Mitglieder haben es mir wirklich leicht gemacht. Danke dafür, falls jemand mitliest. (Was schreibe ich da? Natürlich lest Ihr mit, ist ja schließlich die Vereinszeitschrift :-)

Die Vorträge waren von so hohem Niveau und einer Qualität, dass mir meine Gehirnzellen vollständig weggebrannt wären, hätte ich nicht einen Trick angewendet. Ich habe mir die Ohren zugehalten. Nein, natürlich nicht :-) Ich habe mich auf das Verstehen der Idee und des Konzeptes konzentriert. In die Details gehe ich dann später, wenn mein Wissen entsprechend ist. Ebenso wertvoll waren die Gespräche zwischendurch und am Abend. Was ich dort an Vorschlägen, Tipps und Tricks erhalten habe, gibt meinem Vorankommen einen gewaltigen Schub. Interessiert waren die Mitglieder aber auch an meinen Vorstellungen, Ideen und Verbesserungsvorschlägen. Geben und Nehmen eben.

Zusammengefasst habe ich einen Verein kennengelernt, der aktiv ist und ein unheimliches Potential besitzt. Von Vereinsmeierei übrigens keine Spur. Ich habe bis zum letzten Tag nicht erkennen können, wer überhaupt im Vorstand ist. Forth feiert dieses Jahr 50. Gebutstag. Mit so einem Verein im Rücken sehe ich keine Probleme für die nächsten 50 Jahre. Aber der Verein kann es nicht alleine schaffen. Wenn Ihr so wie ich bisher immer nur heimlich mitgelesen habt, kommt raus aus Euren Löchern und macht mit. Ihr werdet erstaunt sein, welche Möglichkeiten sich auftun. Wer Bedenken hat, sich sofort an den Verein zu wenden und lieber mit einem Frischling plaudert: Meine Kontaktdaten stehen unten.

Ob ich Vereinsmitglied werden möchte? Da kannst Du aber drauf wetten! Ich habe schon zu lange gewartet. Der Antrag ist ausgefüllt und eingetütet. Fehlt nur noch die Briefmarke.

So, ich muss jetzt Schluss machen, die Post schließt gleich...

Wolfgang Strauß — `wost@ewost.de`

... so “browser” will do for now.

Aus der Mottenkiste gefischt von Dirk :-)

> “21st Century Forth Through the group consenses evaluation of Forth and its needs, a basis for a 21st Century Forth can be established. This is what was gleaned from the attendees of the morning session for the Silicon Valley FIG on January 24, **1998**.
>
> Others can add to or improve this list if they like by e-mailing `john.carpenter@stanford.edu` (John D. Carpenter).
>
> This list provides a basis for further SVFIG meeting morning sessions and will ultimately provide input for the next revision of the Forth ANSI standard as the *what* here evolves into the *how* subsequently.
>
> What’s good about Forth: Small, easy to understand, extendable, adaptable to programmer, minimal syntax, interactive, open compiler/interpreter, simple architecture, source code available, helpful in learning about hardware and software.
>
> What’s bad about Forth: NIH (not invented here), cryptic reverse polish notation, difficult to learn, documentation sometimes lacking or unavailable, unconventional syntax, no linkage with other languages, getting more complex, lack of data typing, source (proprietary protection problem).
>
> What 21st Century Forth should be: Work in embedded applications, work with networks and

Internet, work with large systems (operating systems, graphic user interface), be able to be taught to programmers and engineers.

What Forth needs: Reuseable binaries (other languages & headers), more/better/useable libraries, safety (year 2000), better documentation, more examples, workable standards, better gui/window editor w/more information, package confidence, more debugger integration, works with newer CPUs, publications, big daddy $$$, educational opportunities, more types of multitasking/multiprocessing.

21st Century Forth desires and vision real/virtual machines/core, scriptmaking (language to taste), real compile toggle (memory, speed, optimization), integratable applications, 'browser' (*) environment.

Environments (such as voice) interpretation (fuzzy) pda's bases, smart cards, etc servants (robots, etc) applications.

* Note that browser is in quotes; future human interface to a general purpose computer and many dedicated purpose computers may well be derived from what is now a browser interface. There is no name for such an interface yet so 'browser' will do for now."

Quelle: `http://www.forth.org/svfig/21st.html`

TIO — TTY–Terminal IO

Mit TIO gibt es für Linux/Unix ein kleines Terminal–Programm speziell für die Arbeit mit embedded Geräten. Während andere Terminal–Programme sich verabschieden, wenn das Embedded–Gerät bei einem Reset kurz verschwindet, wartet TIO auf das Wiedererscheinen des Anschluss und verbindet sich automatisch wieder mit dem embedded Gerät.

Auf der Webseite zu `tio` gibt es eine kleine Animation, welche die Funktionen des Programms demonstriert.

https://tio.github.io/

Carsten

Latenz von Terminalprogrammen

Die Web-Zeitschrift *Linux Weekly News* `https://lwn.net` hat Terminal-Emulatoren für Linux getestet. Da einige Forth–Entwickler per Terminal–Programm mit Forth sprechen, ist dieses Thema auch hier relevant. Interessant ist aus meiner Sicht der Teil 2, bei dem es um Latenz–Messungen zwischen Tastenanschlag und Erscheinen des Zeichens auf dem Bildschirm geht. Ist die Latenz zu hoch, so mehren sich unbewusste Tippfehler. Aufgrund moderner „Zwiebelprogrammierung" ist die Latenz bei einigen Terminal–Programmen unverhältnismäßig hoch. Das im Sixel–Artikel in diesem Heft erwähnte `mlterm` schneidet zusammen mit `uxterm` recht gut ab. Aber schaut selbst:

`https://lwn.net/Articles/749992/`

`https://lwn.net/Articles/751763/`

Carsten

Micro Launchpad V2

Willem Overkerk hat gerade sein Micro–Launchpad (MLP) herausgebracht. Rund um den MSP430G2553 sind auf dem Plaatje des niederländischen Nachbarn untergebracht ein MCP131–Spannungsregler als Reset–Schaltung, 2 Taster, 1 LED, und als Zugabe sogar ein 64–kBit–I2C–EEPROM. Alle 2x8–Port–Pins sind an den Rand geführt und an der Stirnseite sind die Programmier– und RS232–Anschlüsse sowie die für die Versorgung auf einem 7–Pin–Stecker.

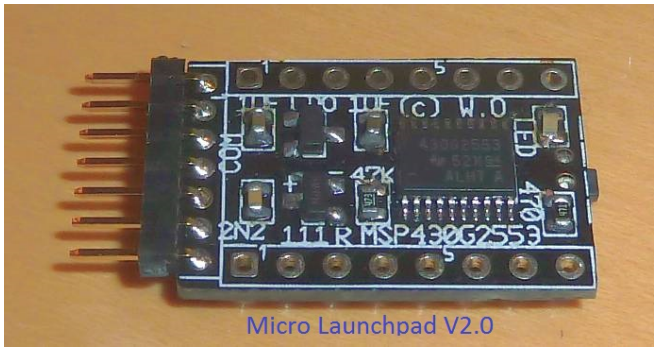


Abbildung 1: Oberseite

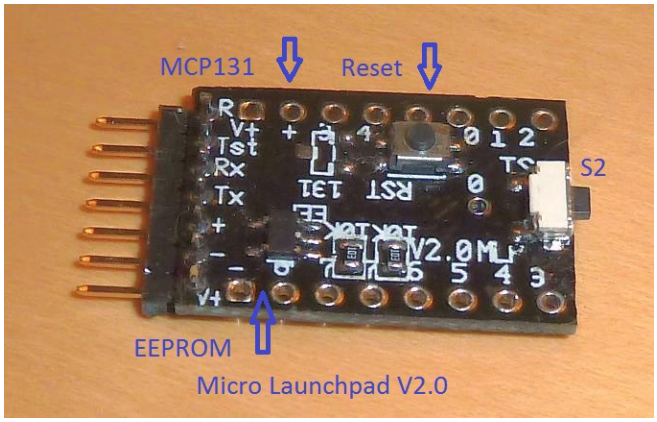


Abbildung 2: Unterseite

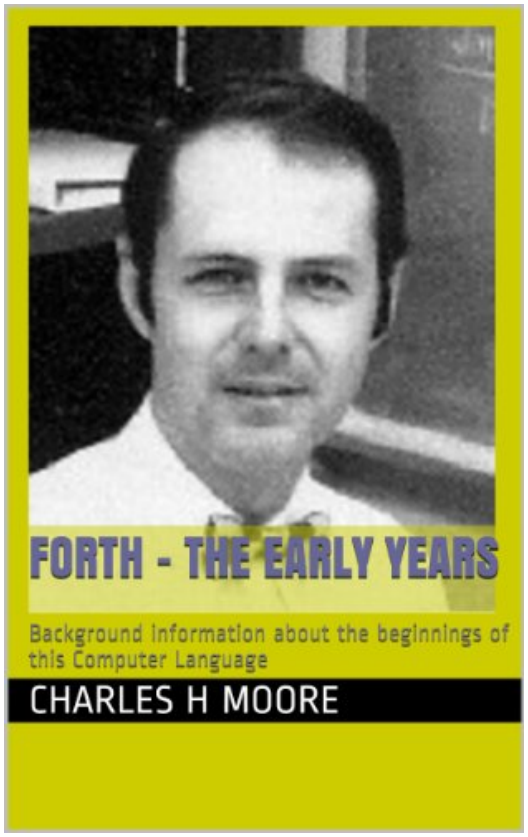
mk

Winguide, Dpans94

Beim Stöbern in Internet habe ich das Programm *winguide.exe* gefunden. Es ist ein Clone des Hypertext–Programms *St–Guide.prg* für den Atari–St. Es wurde geclont von Frank Rüger und läuft auf Windows. Als ich mit dem Atari–St arbeitete, schrieb ich einen Hypertext vom Dpans94. Es kann jetzt mit Winguide gelesen werden. Die Dateien *winguide.exe* und die **.hyp–Dateien* können in einen beliebigen Ordner gelegt werden, im *liesmich.txt* ist das erklärt. Ich habe mir dafür einen neuen Ordner C: \Programme\Winguide gemacht und alles dareingelegt. Legen sie eine Verknüpfung zu *dpans.hyp* auf dem Desktop an. Rechts–Klick auf *dpans.hyp* und in „Öffne

mit ...“ die *winguide.exe* suchen und öffnen. Sie werden von der Anzeige und deren Inhalt sehr begeistert sein.
Im Menü „Datei“ gibt es unter anderem den interessanten Eintrag „Recompilieren ... R-, “. Es liefert die Quelldatei *dpans.stg,* die mit einem Editor verarbeitet werden kann. Leider darf man sie nicht compilieren. Es wäre zwar möglich, doch der Autor von St–Guide, Holger Weets, hat es nicht erlaubt. Lesen sie bitte hierfür die Datei *changes.txt.* Es ist wirklich schade, denn durch Hinzufügen von Worten von Forth–2012 könnte man die Datei *forth2012.stg* erzeugen, sie compilieren und *forth2012.hyp* bekommen. Mit Winguide geht es nicht zu compilieren, aber es gibt einen Ausweg. Im nächstem Leserbrief werde ich darüber berichten. Die Datei *hcp.hyp* liefert die Dokumentation zu den **.stg*–Dateien und wie man sie herstellen kann. Die Dateien in der Anlage[1] dürfen weitergegeben werden.
Herzliche Grüße, Filippo Sala

P.S Der Beitrag *Instruktionen für die Compilation von *.stg-Dateien* — der Ausweg — wird bald fertig sein. Wenn jemand jetzt schon interessiert ist, bitte per E–mail mitteilen: `filippo.sala@gmx.de`, und ich werde ihm den Beitrag per E–mail schicken.



Forth is a very special computer language — as is the inventor Charles H Moore. This eBook describes the early years of Forth, a simple natural computer language as Charles describes it, and how it developed over the years. It documents the different locations where the inventor Charles H Moore worked and optimized Forth. MIT 1958, Stanford 1961, Freelance 1965, Mohasco 1968, NRAO 1971.

Within programming community there is a very strong feeling about this language — for it and against it, often quite emotional. Charles' moral from the eBook: I know Forth is the strongest language so far. I'm disturbed that people who should don't appreciate how it embodies their own description of the ideal programming language. Forth has lead to an architecture that promises a wonderful integration of software and silicon. One way it has been described: You need longer to learn it to work with it efficiently — but from then on you will program faster. The same even seems to apply if the programmes steps back to her/his language of choice. [Kindle Edition; by Charles H Moore (Author), Juergen Pintaske (Editor); https://www.amazon.co.uk] mk

A Brief History of Hard Drives

1956

In 1956, ... IBM introduced the first iteration of the piece of technology we know as the hard disk drive. The 305 RAMAC system came equipped with fifty 24-inch platters and had a total capacity of 5MB. Using only a single read/write assembly sporting two heads in order to access each platter, the 305 RAMAC had an access time of nearly one second. It was also about the size of two household refrigerators, side by side. ...

Mit dem Netz gefischt:

https://www.gillware.com/blog/articles/a-brief-history-of-hard-drives/

mk

[1] Die Anlage hat Bernd für euch hinterlegt als `https://forth-ev.de/~bp/dpans94.zip`

"Code–it–yourself" Manifesto

Gefunden unter `https://pestilenz.org/~ckeen/blog/posts/ciy-manifesto.html`

We use software for our everyday needs because we want to get something done. We have goals to achieve and things to do.

The software we use is coded by brave programmers that have their own goals. Most of the time there is an overlap between their goals and ours.

Over time these will diverge.

This means that the tools we depend on grow features we don't use or understand. There will be bugs in these code parts which will prevent us from reaching our goals.

So we are at a fork in the road:

- We have the choice of trying to understand the code and fix it.
- We have the choice of trying another program, whose creator's goals are closer to ours.
- We also have the choice of coding the software ourself.

All but the last path mean endless seeking, evaluating and further deviation from our goals. Therefore we replace programs we do not understand fully with our own implementation.

The followers of the Code It Yourself Manifesto believe in these things:

- We implement it according to our own goals.
- We make mistakes and learn from them.
- We learn how our tools we depend on need to work.
- We gain a deep understanding of our problem domain.

We still embrace sharing of ideas and code.

Sharing is only possible if we are excellent developers to each other. The next developer reading our code will be us in a not so distant future. Coding It Ourselves means we will document our code, clearly stating the goal of the software we write.

Together we enjoy the diversity of implementations and ideas.

We encourage our colleagues to

Code It Yourself.

Gforth im Container

Matthias Trute

Carsten hat vor einiger Zeit [1] gforth als Prozess 1 für Linux vorgestellt. Das hat Hack–Value. Man kann gforth auch im Docker Container laufen lassen, das ist fast noch cooler. Weil, da brauchts keinen eignen Linux–Kernel mehr, es geht ohne Neustarts und Zugriff auf Dateien gibt es obendrauf.

Wie sieht das aus

Nimm ein beliebiges PC Linux, Windows 10 ist vielleicht auch ok. Nimm den Anwendungsmanager deiner Wahl und installiere `docker`. Manchmal heißt das Package auch `docker.io`. Beispielhaft Ubuntu/Debian:

```
$ sudo apt-get install docker.io
$ sudo adduser ich docker
```

Dann einmal neu anmelden, damit der aktuelle User `ich` auch Mitglied der neuen Gruppe `docker` wird, die bei der Installation eingerichtet wird. Das geht fix. Dann öffne ein Kommandofenster und starte gforth

```
$ docker run -ti mtrute/gforth-container
Unable to find image 'mtrute/gfo...
latest: Pulling from mtrute/gfor...
ff3a5c916c92: Pull complete
e80f99756aa5: Pull complete
Digest: sha256:60ee2a644af2c ...
Status: Downloaded newer image ...
Gforth 0.7.9_20180208, Copyright..
Gforth comes with ABSOLUTELY NO ..
Type 'help' for basic help
```

Die Zeilen sind abgeschnitten, damit es lesbar bleibt. Das wars schon. Die üblichen Befehle wie `WORDS` funktionieren. Auch sonst ist alles, wie man es kennt. Ende der Session mit `BYE`. Und alles ist weg. Auch wie sonst.

Was ist also anders? Gforth kam „aus der Cloud“ als etwa 20MB großer Download. Und es ist brandaktuell[1]. Und es hat rein gar nichts mit einem eventuellen anderen gforth auf dem Rechner zu tun.

Theorie

So etwa 2015 kam der Hype namens Docker auf. Waren bis dahin vor allem virtuelle Maschinen (VM) in aller Munde, verschiebt sich seither der Fokus langsam in Richtung Anwendungen. Der Grund ist eigentlich einfach: Meist will man eine oder einige wenige Anwendungen in seiner VM laufen lassen. Zum Beispiel einen Webserver mit Nextcloud für seine Daten samt Kalender für das Handy, ohne dass Google mitliest und Werbung passend zum Zahnarzttermin schaltet. Mit einer eigenen VM kein Problem, sie hat nur den Nachteil, dass sie außer Nextcloud und dem Webserver noch ein komplettes Betriebssystem mitbringt, was auf Dauer ziemlich lästig werden kann.

Der zweite Punkt war, und ist, dass es Anwendungen gibt, die partout nicht mit anderen Anwendungen zusammen laufen können. Webserver kämpfen seit jeher mit den Versionen von PHP oder Java und deren Bibliotheken. Dauerhaft funktioniert das eigentlich nur, wenn jede Anwendung auf einem eigenen Server läuft. Das ist auf Dauer noch lästiger und teuer obendrein.

Mit Containern kam das Versprechen, dass das alles der Vergangenheit angehört. In einem Container wird eine Anwendung mitsamt allen Bibliotheken und Modulen verpackt. Wird der Container gestartet, wird die Anwendung gestartet und verhält sich so, als ob sie direkt gestartet ist. Der Clou ist, dass die Module und was sonst noch alles verpackt wurde, nur für diese Anwendung verfügbar sind. Außerhalb des Containers ist nichts davon zu bemerken. Technisch gibt es das schon lange, Unix kennt das `chroot`–Konzept gefühlt so lange, wie es Forth gibt, und die Freunde der Berkeley Unixe (BSD) haben ihre Jails nur unwesentlich weniger lang. Die Leute bei Docker haben die Ideen „einfach“ weiter gedacht, ein paar Werkzeuge drumherum gebaut und das Ergebnis Container genannt. Die Linuxer haben dazu passend ein paar Dinge bereitgestellt, damit das hübsch geschmeidig laufen kann. Sogar Microsoft hat ziemlich schnell die Idee adaptiert und in Windows 10 eingebaut. Soll ganz gut laufen, habe ich gehört.

Die Firma hinter Docker betreibt daneben einen Dienst, in dem man ohne Weiteres Anwendungen einstellen kann: Docker Hub [2]. Wenn man für diesen Dienst nichts bezahlt, sind alle Anwendungen, die man dort einstellt, automatisch für alle verfügbar. Das lokale Docker aus der obigen Installation ist so voreingestellt, dass es automatisch dort suchen kann.

```
$ docker search gforth
NAME                        DESCRIPTION   STARS
jwodder/gforth              A fast and    2
druffer/gforth              Run gforth    1
mtrute/gforth-container     gforth in a   0
oldmankris/gforth           gforth on t   0
oldmankris/alpine-gforth    Gforth on a   0
idomyk/gforth-mac           gforth mac    0
```

Praxis

Wie funktioniert das nun? Am Anfang steht ein Textfile. Das heißt einfach `Dockerfile` und liegt in einem eigenen Verzeichnis mit beliebigem Namen. In diesem File steht, was wie zu tun ist, um eine Anwendung zu installieren

[1] Dieser Artikel ist im Wesentlichen bereits vor Ostern entstanden, die Versionsnummern sind bei Erscheinen des Artikels mit Sicherheit nicht mehr aktuell.

und wie das Programm heißt, das gestartet werden soll, wenn der Container aufgerufen wird.

Als Beispiel zum Warmwerden nehmen wir gforth. Die Linuxe dieser Welt sind schrecklich weit hinter dem Stand der Dinge hinterher. Am Schlimmsten ist, dass alle diese gforth's keine Recognizer kennen. Also nehmen wir den aktuellen Stand von Antons Webserver und bauen gforth selbst. Das klingt schrecklich und das ist es auch. Denn es müssen sehr viele Pakete vorhanden sein, damit gforth mit allen Features entstehen kann. Hat man einmal gforth gebaut, braucht man diese Pakete aber nicht mehr und könnte sie eigentlich wieder löschen, das nächste Update wird dadurch nur umfangreicher.

Mit dem nachfolgenden `Dockerfile` wird der gesamte Bau–Prozess automatisiert. Das dauert auch auf schnellen Rechnern durchaus die eine oder andere Minute, man kann jedoch meist entspannt zusehen, ein paar Warnungen und schlimm klingende Meldungen vermeiden Langeweile zuverlässig.

```
FROM alpine:3.7
ENV LANG C.UTF-8

ENV VERSION 0.7.9_20180208
ENV URL https://www...../forth/gforth/Snapshots

RUN  apk add --no-cache libltdl libffi \
    && apk add --no-cache --virtual .fetch-deps \
        build-base wget m4 libtool file xz tar \
    && wget $URL/$VERSION/gforth-$VERSION.tar.xz \
        -O /tmp/gforth.tar.xz \
    && xzcat /tmp/gforth.tar.xz | tar xf - \
       -C /tmp  \
    && rm /tmp/gforth.tar.xz \
    && cd /tmp/gforth-* \
    && apk add --no-cache --virtual .build-deps \
        freetype-dev coreutils gcc swig \
        libffi-dev mesa-gles mesa-dev libx11-dev \
    && ./configure --prefix=/usr \
    && make \
    && make install \
    && cd /tmp && rm -rf gforth-* \
    && apk del .build-deps \
    && apk del .fetch-deps

CMD [ "gforth" ]
```

Die erste Zeile markiert die Grundlage, auf der wir arbeiten. Alpine ist ein bis auf das absolute Minimum abgespecktes System, das grade mal so ein Programm starten kann. Dafür ist es auch nur sehr wenige MB (4 derzeit) groß. Es kann gar nichts. Aber es hat einen Paketmanager, der auf einen ziemlich gut gefüllten Bestand zugreifen kann.

Mit diesem Paketmanager, apk, laden wir nun die vielen Module, die gforth für das Compilieren braucht. Das sind mit den ganzen Abhängigkeiten so um die 100. Dann holt wget das gforth Sourcepaket in Wien ab. Der Rest folgt dem, was in `BUILD-FROM-SCRATCH` steht. Am Ende wird sauber aufgeräumt und es bleibt ein Paket von etwas über 20 MB Größe übrig.

```
$ docker build -t gforth .
Step 1/6 : FROM alpine:3.7
 ---> 3fd9065eaf02
Step 2/6 : ENV LANG C.UTF-8
 ---> Using cache
 ---> 64366f3611ad
Step 3/6 : ENV VERSION 0.7.9_20180208
 ---> Using cache
 ---> 5112d9e4337a
Step 4/6 : RUN adduser

... und noch viele viele weitere Zeilen

(31/32) Purging libmagic (5.32-r0)
(32/32) Purging xz-libs (5.2.3-r1)
Executing busybox-1.27.2-r7.trigger
OK: 4 MiB in 13 packages
 ---> 27f16467ac1f
Removing intermediate container 50f25d7c0bc2
Step 5/6 : USER gforth
 ---> Running in 9580bf593522
 ---> 3d25db7a80bf
Removing intermediate container 9580bf593522
Step 6/6 : CMD gforth
 ---> Running in bb5da4c99701
 ---> 85b312985829
Removing intermediate container bb5da4c99701
Successfully built 85b312985829
$ docker images
REPOSITORY  TAG     IMAGE ID      SIZE
gforth      latest  85b312985829  21.5 MB
```

Das geht im Prinzip auch ohne Container. Der apk hat aber den Charme, dass er die zusätzlichen Pakete, die er installiert, abspeichern kann und später diese Liste nehmen kann, um aufzuräumen. In dem obigen File ist dies der Parameter `--virtual`, der, wie alle des Englischen Mächtigen wissen, nicht imaginär oder eingebildet heißt, sondern einfach praktisch.

Wer mag oder muss, kann auch die sonst üblichen Verdächtigen als Grundlage nehmen. Debian, Ubuntu, Fedora und sogar ein Microsoft Windows Nano Server sind als Docker Container verfügbar und können genutzt werden. Dann sind die Container aber deutlich größer: gforth auf Debian hat einige Hundert MB am Ende auf der Waage. Ohne mehr zu können. Außerdem muss man natürlich auch anstelle des apk auch die apt's und yum's dieser Welt nutzen.

Mit diesem selbst gebauten Container kann man nun zum Docker Hub Service gehen und ihn hochladen. Ganz klassisch. Cooler ist aber, wenn man nicht die fertigen, unter Umständen doch voluminösen Container hochlädt, auch wenn das heutzutage eher banal erscheint, sondern nur das kleine Dockerfile. Das geht über github.com (3). Dort registrierte Projekte können sich mit Docker Hub verlinken und jede Änderung bei github löst automatisch einen Rebuild bei Docker Hub aus. Weil auch Docker das toll findet, ist die Spalte „Automated Build“ beim `docker search` Befehl eingeblendet. Da steht ein `[OK]` immer dann, wenn der Container anhand der Anweisungen aus github bei Docker Hub entstanden ist.

Richtige Praxis

Mit dem obigen Container kann man schon mal spielen.

```
$ docker run -ti gforth
Gforth 0.7.9_20180208, Copyright (C) 1995-2...
Gforth comes with ABSOLUTELY NO WARRANTY; f...
Type ‘help’ for basic help
(Cursor blinkt)
```

Man kann Worte definieren und ausführen. Definieren ist hier sehr traditionell: Eintippen. Nicht aus Dateien einlesen. Wenn man aber doch ein `s" file.fs" included` probiert, passiert Folgendes:

```
s" file.fs" included
*the terminal*:1: error: No such file or directory
s" file.fs" included
Backtrace:
     0 $7F3FAB43D720 throw
```

Warum ist das so? Eine Idee von den Containern ist, dass sie maximal abgeschottet sind. Dazu gehört auch der Zugriff auf das Filesystem und auf Dateien. Das wird schlicht unterbunden. Wie kann man nun doch Dateien in den Container reinbekommen? Zwei Wege: Der erste ist, sie in den laufenden Container hineinzukopieren. Dazu braucht man ein zweites Kommandofenster. In der ermittelt man zuerst die ID des laufenden Containers, mit dem man Dateien austauschen will.

```
$ docker ps
CONTAINER ID        IMAGE                       COMMAND
410e818740dd        mtrute/gforth-container     "gforth"
```

Mit dieser ID ist es nun ein Leichtes, Dateien zu kopieren. Das Beispiel gibt beim Laden eine Meldung aus, macht aber sonst nichts.

```
$ cat test.fs
.( huhu von test.fs )

$ docker cp file.fs 410e818740dd:/tmp
```

Ein `/tmp` gibt es tatsächlich im Container, aber ein ganz anderes als das globale `/tmp`. Von dort ist das Laden des Files eine einfache Fingerübung

```
s" /tmp/file.fs" included huhu von test.fs ok
```

Mehrere Dateien gehen auch. Es wäre allerdings doch schöner, ein Verzeichnis in den Container einzublenden um dann von dort zu lesen. Das geht mit dem gforth–Container von Docker Hub nicht. Das ist aber nicht schlimm. Container sind wie russische Matroska Puppen: Man kann sie ineinanderstecken. Also schaffen wir uns einen eigenen Container, der genau das macht, was wir wollen.

Start ist wie immer ein `Dockerfile` in einem leeren Verzeichnis. Diesmal ein sehr einfaches, in dem nur ein Verzeichnis `/Daten` definiert wird.

```
FROM mtrute/gforth-container
VOLUME /Daten
WORKDIR /Daten
```

Daraus bauen wir uns unseren eigenen Container. Das dauert diesmal nur Sekunden und braucht praktisch keinen Speicherplatz.

```
$ docker build -t mein-gforth .
Sending build context to Docker daemon 2.048 kB
Step 1/3 : FROM mtrute/gforth-container
 ---> 099e82a1b71c
Step 2/3 : VOLUME /Daten
 ---> Using cache
 ---> 0aa2cfe2d6b5
Step 3/3 : WORKDIR /Daten
 ---> Using cache
 ---> 1677f88784fb
Successfully built 1677f88784fb
```

Beim Start des Containers kann man jetzt angeben, wie das so definierte Verzeichnis mit der Außenwelt verbunden sein soll. Macht man dies nicht, hat man nur ein leeres Verzeichnis.

```
$ docker run -ti -v /home/ich:/Daten -w /Daten mein-gforth
```

Damit wird der gesamte Verzeichnisbaum unter `/home/ich` unter dem Namen `/Daten` im `mein-gforth` Container eingebaut und man kann auf `/home/ich/test.fs` unter dem Namen `/Daten/test.fs` zugreifen. Änderungen an `/home/ich/test.fs` sind sofort im Container sichtbar. Das funktioniert so ähnlich wie ein NFS Mount oder ein Netzlaufwerk.

```
$ docker run -ti -v /home/mt:/Daten -w /Daten mein-gforth
Gforth 0.7.9_20180208, Copyright (C) 1995-2...
Gforth comes with ABSOLUTELY NO WARRANTY; f...
Type ‘help’ for basic help
s" /Daten/test.fs" included huhu von test.fs  ok
```

Dies und Das

Anlass war Bernds net2o. Das lief bei mir zuerst auf einer VM, die 2 GB RAM von meinem Server weggeknabbert hat. Seinerzeit nicht schlimm, weil sowas halte ich für normal. Irgendwann kam Ubuntu mit, allerdings eigenen, Containern um die Ecke. Damit waren's nur noch wenige Hundert MB. Schon schick. Aber der Server wurde alt und verbrauchte mehr Strom, als mittlerweile vertretbar war. Die Entscheidung fiel auf eine NAS Box. Die ist viel kleiner, leiser und braucht nur einen Bruchteil Energie. Die brachte Docker mitsamt einer hübschen Browser–GUI mit. Erste Experimente damit waren ermutigend, so dass jetzt das gesamte nerdige Smart–Home Zeugs nicht mehr verteilt auf Server und Raspberry–Pi's sondern in einem Containerset läuft.

Genauso einfach war es, Bernds net2o–Pakete zu nutzen. Die basieren auf Ubuntu bzw Debian und die gibt es auch als Container.

```
FROM ubuntu:latest

ENV DEBIAN_FRONTEND noninteractive
ENV LANG C.UTF-8

RUN apt-get update \
 && apt-get install -y gnupg apt-utils\
     software-properties-common \
```

```
     apt-transport-https curl \
 && apt-add-repository \
   'deb https://net2o.de/debian testing main' \
 && curl -L https://net2o.de/\
bernd@net2o.de-yubikey.pgp.asc \
   | apt-key add - \
 && apt-get update \
 && apt-get install -y net2o \
 && apt-get autoremove -y && apt-get autoclean

CMD [ "n2o" ]
```

Im net2o Chat haben die Forthiker auch prompt gelästert, wie dick und fett denn das alles sei. In der Tat wollte ich eher ein generisches gforth haben, mit dem man *auch* ein net2o laufen lassen kann. Bernds Lösung, ein embedded gforth nur für net2o zu haben, ist pragmatisch, aber mir zu retro, sowas machen Java–Leute und das seit Jahrzehnten. Die Arbeit an dieser Idee ist noch nicht ganz fertig, nur ein paar Grafikbibliotheken fehlen noch. Alpine hat viel, aber nicht alles.

Gforth wird wie net2o rege entwickelt. Eine bestimmte net2o–Version erfordert dabei fast immer auch eine bestimmte gforth–Version. So funktioniert net2o–Version `0.6.0-20180307` vermutlich nur mit gforth–Version `0.7.9_20180208`. Das bekommt man hin, indem die Docker–Container einen Tag erhalten, das der Versionsnummer entspricht. Bernd hat im net2o–Fossil die Versionen ebenso markiert. Ein Dockerfile für net2o sieht dann so aus:

```
FROM mtrute/gforth-container:0.7.9_20180208

RUN apk add --no-cache --virtual .build-deps \
  fossil git build-base m4 libtool file \
  libffi-dev libltdl g++ mesa-dev libx11-dev \
  autoconf automake pcre-dev bison \
  boost zlib-dev coreutils mesa-gles \
 && mkdir /tmp/net2o-src \
 && cd /tmp/net2o-src \
 && fossil clone https://fossil.net2o.de/net2o \
  net2o.fossil \
 && fossil open net2o.fossil 0.6.0-20180307 \

usw
```

Daneben gibt es auch das Tag `latest`. Es verweist immer auf die aktuelle Version und ist auch der Default, wenn kein Tag angegeben ist. Im Fossil ist die Entsprechung `trunk`. Docker prüft aber nicht selbst, ob es Updates gibt. Dies macht man sinnvollerweise ohnehin selbst, anderenfalls wüsste Docker ja auch von jedem Start eines Containers irgendwo auf dieser Welt.

```
$ docker run -ti mtrute/gforth-container
Gforth 0.7.9_20180208, Copyright (C) 1995-2016,2017 Fr
Gforth comes with ABSOLUTELY NO WARRANTY; for details
Type ‘help’ for basic help
bye

$ docker pull mtrute/gforth-container
latest: Pulling from mtrute/gforth-container
ff3a5c916c92: Already exists
49684fd5c659: Pull complete
Digest: sha256:462f4c7e71f7706d7b7e084afd70aae6ef868df
Status: Downloaded newer image for mtrute/gforth-conta

$ docker run -ti mtrute/gforth-container
Gforth 0.7.9_20180329, Copyright (C) 1995-2017 Free So
Gforth comes with ABSOLUTELY NO WARRANTY; for details
Type ‘help’ for basic help
bye
$
```

Einfacher kann man nicht up-to-date bleiben. Eigene Container sind so schnell aktualisiert wie sie beim ersten Mal entstanden sind, die eigenen `Dockerfile`'s also nicht wegwerfen.

Apropos Updates. Die reine Lehre sagt, Updates geschehen indem man den Container austauscht. Das stimmt sogar. Auf seine Daten muss man natürlich aufpassen, aber die sind ja nicht im Container sondern werden dort nur mit der Option `-v` eingeblendet. Hat alles geklappt, löscht man den alten Container und ist fertig. Wenn nicht, startet man den alten einfach wieder. Schlimmstenfalls muss man das Verzeichnis aus dem Backup oder was auch immer für die Sicherung genutzt wurde, zurückholen.

Es gibt noch eine Eigenheit. Das Archiv von Antons Server ist nicht so ganz ein Abbild des git Sourcerepositorys von Savannah mit den reinen Quellen. Denn eigentlich würde man den Container aus den ultimativen Quellen bauen wollen. Der Grund ist einfach: gforth braucht ein, gerne auch total altes, gforth um sich selbst zu übersetzen. Da im alpine System kein gforth vorhanden ist, funktionieren die Savannah–Quellen nicht. Was das alte gforth macht, ist in Antons tar File enthalten, daher klappt das damit.

Verweise

1. VD 2014-02, S.27ff — C. Strotmann, Forth als PID 1
2. `https://hub.docker.com/r/mtrute/gforth-container/`
3. `https://github.com/mtrute`
4. `https://savannah.gnu.org/git/?group=gforth`
5. `https://www.complang.tuwien.ac.at/forth/gforth/Snapshots/`

Sechs Pixel gehen in eine Bar und bestellen ein Sixel . . .

Carsten Strotmann

Schon seit einiger Zeit war ich auf der Suche nach einer Grafikausgabe, welche auf verschiedenen Betriebssystemen funktioniert und ohne spezielle Schnittstellen auf der Forth–Seite auskommt. Die hier vorgestellte Sixel–Grafik bietet genau diese Funktionen.

Grafik im Terminal — Warum?

Dieser Artikel ist der erste in einer losen Reihe zum Thema „Grafikausgabe im Terminal“. Ein Computer–Terminal verbindet man in der Regel nicht mit Grafik, sind Terminals doch die Nachfolger der TTYs, der Teletypes (Fernschreiber). Jedoch gibt es einige Terminal–Programme unter Windows, MacOS und Linux/Unix, die Grafikausgabe erlauben.

Aber warum sollte man Grafik im Terminal ausgeben? Dafür gibt es mehrere Gründe:

- Grafikausgabe im Terminal ist über Escape–Codes realisiert (spezielle Zeichen, welche nicht gedruckt werden, sondern die Funktion des Terminals verändern). Diese Escape–Codes werden über die normale Zeichenausgabe–Schnittstelle des Betriebssystems realisiert. Bei Forth ist das `emit`. Und ein `emit` hat (fast) jedes Forth, und dort, wo es fehlt, ist es trivial, ein solches Wort hinzuzufügen.

- Diese Art der Grafik ist Betriebssystem–unabhängig, Terminal–Programme mit Sixel gibt es für Windows, MacOS, Linux/Unix und Android–Systeme

- Die Schnittstelle über Escape–Codes ist leichtgewichtig und hat keine Abhängigkeiten zu Programmier–Bibliotheken und komplexen Grafik–APIs. Terminal–Grafik funktioniert auch von einem kleinen embedded Forth–System, welches über USB oder RS232 angeschlossen ist. Damit ist eine grafische Visualisierung von Daten auch von kleisten Forth–Systemen aus möglich.

- Grafik im Terminal kann über die normalen Möglichkeiten in einer Terminal–Kommandozeile manipuliert werden. So kann die Grafikausgabe des Forth–Systems in eine Datei umgeleitet werden (um das Grafik abzuspeichern) oder per Pipe in einen Filter zur weiteren Bearbeitung übergeben werden. Hilfsprogramme ermöglichen es, die Grafikdaten in übliche Grafikformate wie PNG, JPEG oder GIF umzuwandeln.

Die Beispielprogramme in diesem Artikel sind mit Hans Bezemers *4th*–Forth erstellt, sollten aber ohne große Änderungen auch auf andere Forth–Systeme portierbar sein.

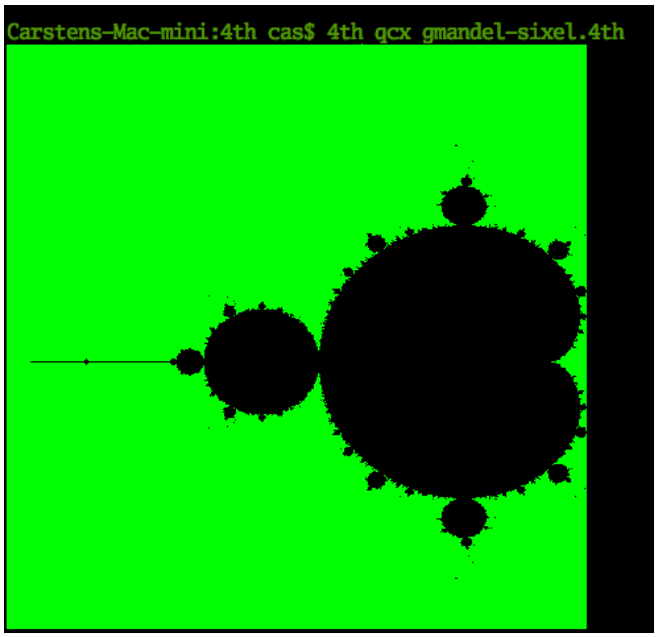


Abbildung 1: Mandelbrot–Sixel–Grafik

Sixel–Grafik

Sixel Grafik wurde in den 1970er Jahren zuerst für Drucker entwickelt und später für die bekannten DEC–Terminals (z.B. VT420) angepasst. Ursprünglich war Sixel–Grafik nur monochrom (also schwarz und weiß) verfügbar, später konnten mit neuen Control–Zeichen die „Farbbänder“ ausgewechselt werden, um mehrere Farben „übereinander“ zu drucken. Heute wird Sixel–Grafik von Terminal–Programmen angezeigt, das „übereinander“ Drucken von Farben funktioniert immer noch.

Wie Sixel funktioniert

Sixel–Modus anschalten

Durch die Ausgabe der Zeichen–Sequenz `ESC-P-q` wird der Sixel–Modus eines sixel–fähigen Terminals angeschaltet. Zwischen dem `P` und dem Zeichen `q` können noch weitere Parameter angegeben werden, z.B. die Geometrie eines Pixels des Ausgabegeräts (Aspect–Ratio), so dass die Grafik nicht verzerrt ausgegeben wird. Dies ist bei der Ausgabe auf ein Terminal nur in Ausnahmefällen notwendig. Informationen zu den Details der erweiterten Parametern gibt es in den am Ende des Artikels angegebenen Dokumenten.

```
DECIMAL

27 emit char P emit char q emit
```

Sixel–Modus ausschalten

Durch die Ausgabe des Zeichens mit dem dezimalen Wert `156` wird der Sixel–Modus des Terminals wieder ausgeschaltet und in den normalen Terminal–Modus (Text) zurückgeschaltet. Durch an- und ausschalten des Sixel–Modus lassen sich Text und Grafik mischen. Oft bewirkt auch die Ausgabe eines Zeichens, welches nicht als Sixel–Befehl interpretiert werden kann, einen Abbruch des Sixel–Modus.

Sixel ausgeben

Die Werte zwischen `63` und `127` erzeugen im Sixel-Modus ein vertikales Pixel–Muster von sechs Pixeln untereinander (six pixel = sixel). Das oberste Pixel hat den Wert `1`, das sechste Pixel unten den Wert `32`. Alle Pixel–Werte werde addiert und es wird die Konstante `63` zur Summe hinzuaddiert, um in den Bereich der druckbaren ASCII–Zeichen zu kommen.

```
 1  | |
 2  |X|
 4  |X|
 8  | |
16  |X|
32  | |
=       22 + 63 == 85
```

Unter Forth kann der errechnete Wert mit dem Wort `emit` ausgegeben werden:

```
85 emit
```

Es gibt keine Längenbegrenzung der Sixel–Zeile. Daten, welche über den rechten Rand des Terminals heraus geschrieben wurden, kommen nicht zur Anzeige.

Abbildung 2: Stumble–Labyrinth–Grafik via Sixel

Sixel–Steuerzeichen

Es gibt eine Reihe von Werten welche im Sixel–Mode als Steuerzeichen interpretiert werden:

- **$** positioniert den Sixel–Cursor an den Anfang der aktuellen Zeile, ohne die Zeile dabei zu wechseln (klassisches „Carriage–Return“ Wagenrücklauf **ohne** Linefeed). Dies wird benötigt, um mehrfarbige Bilder zu erstellen. Für mehrfarbige Bilder wird vor der Ausgabe der ersten Zeile die erste Farbe eingestellt und die Pixel dieser Farbe ausgegeben. Danach folgt der Wagenrücklauf und der Wechsel der Farbe, in welcher dann die nächsten Pixel ausgegeben werden. Moderne Terminal–Emulatoren unterstützen 4/16/256/32tsd Farben (Dokumentation zum Terminal–Programm beachten).
- Das Zeichen **-** sorgt für einen Zeilenvorschub. Bei einigen Terminal–Emulatoren wird dabei auch der Cursor auf den Anfang der Zeile gesetzt. Es wird empfohlen, ein **-** immer zusammen mit **$** auszugeben.

Grafik im Speicher als Sixel–Grafik ausgeben

Das nachfolgende Forth–Wort gibt eine Grafik per Sixel auf einem Terminal aus. Die Variablen `pic_height` und `pic_width` beinhalten die Größe des Bildes in Pixel, `pixel@` liefert den Pixelwert an einer Koordinate zurück. In diesem Beispiel ist die Ausgabe monochrom, ein Wert in einer Speicherstelle bedeutet, dass ein Pixel gesetzt ist (Diese Datenstruktur lässt sich speichersparender Implementieren, diese Implementierung ist sehr vereinfacht).

```
: SIXEL-DUMP
  pic_height @ 0 DO
    pic_width @ 0 DO
       SIXEL-VAL OFF
       j i  6 0 DO
OVER OVER
SWAP i 1- + SWAP
pixel@ IF  1 i LSHIFT  SIXEL-VAL +! THEN
      LOOP
      DROP DROP
      SIXEL-VAL @ 63 + emit ( SIXEL ausgeben )
    LOOP
    CHAR - emit ( SIXEL CR/LF )
  6 +LOOP ;
```

Sixel farbig

Sixel–Ausgabegeräte besitzen Farbregister. Diese Farbregister enthalten beliebige Farben welche per RGB oder HLS definiert werden können. Das Sixel–Steuerzeichen # gefolgt von einem Wert wählt ein Farbregister (0 — 255).

Wird ein # gefolgt von einer Zahl gefolgt von einem Semikolon ; ausgegeben, so folgt danach eine Farbdefinition als R/G/B Werte jeweils getrennt durch Semikolon.

Das folgende Beispiel definiert die Register 0/1/2 per RGB Farb–Werte neu:

```
#0;2;0;0;0#1;2;100;100;0#2;2;0;100;0
```

Möchte man neue Farben in der HLS–Notation eingeben, so kann HLS beim Anschalten des Sixel–Modus auswählen.

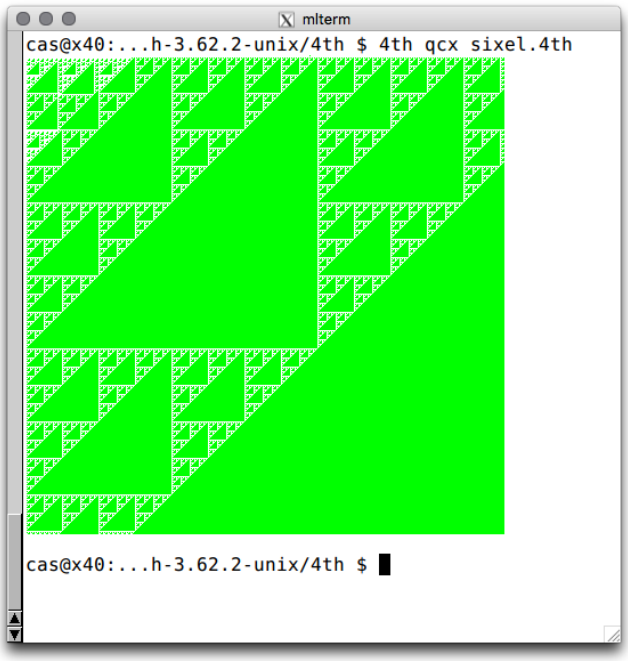


Abbildung 3: Sierpinski–Dreiecke via Sixel

Sixel–Grafiken speichern

Da Sixel–Grafiken normale Textausgaben sind, können diese ohne Probleme auf der Kommandozeile per Ausgabeumleitung in eine Datei geschrieben werden. Dieses Beispiel speichert die Ausgabe eines Forth–Programms in die Datei `mandelbrot.six`:

```
# 4th cxq gmandel-sixel.4th > mandelbrot.six
# file mandelbrot.six
mandelbrot.six: Non-ISO extended-ASCII text,
   with very long lines,
   with no line terminators,
   with escape sequences
```

Die Grafik kann jederzeit unabhängig vom Forth–Programm wieder anzeigt werden. Unter Unix/Linux per `cat`, unter Windows per `type`:

```
# cat mandelbrot.six
```

Programme wie der Graphics–Converter (MacOS X) können Sixel–Grafiken einlesen und in anderen Formaten (PNG, GIF etc) wieder speichern.

Sixel einsetzen

Den Forth–Quellcode rund um die Sixel–Grafik habe ich auf GitHub hinterlegt — Links. Die Programme sind benutzbar, benutzen aber noch nicht alle Möglichkeiten der Sixel–Grafik. Es fehlen noch

- Unterstützung des Run–Length–Encoding (RLE) innerhalb der Sixel–Daten
- Sixel–Bibliothek für ANSI–Forth/Forth–200x
- Forth–Wörter für die Farbausgabe
- ein Paket für TheForth.net

Wer mit Sixel arbeitet und eigenen Quellcode zur Verfügung stellen möchte, kann per GitHub einen Pullrequest senden (oder ganz traditionell den Quellcode per E–Mail).

Terminal–Emulatoren mit Sixel–Unterstützung

Eine unvollständige Liste der Terminal–Emulatoren mit Sixel–Unterstützung

- Windows
  - mlterm (`https://mlterm.sf.net`)
  - rlogin (Japanisch `https://mintty.github.io/`)
  - mintty (`https://mintty.github.io/`)
- MacOS X/Unix/Linux
  - mlterm (`https://mlterm.sf.net`)
  - xterm (muss selbst mit Option -enable-sixel"compiliert werden)
- Android
  - mlterm (`https://mlterm.sf.net`)
- MS–DOS
  - Kermit (`http://www.columbia.edu/kermit/mskermit.html`)

Sixel–Dokumentation

- Wikipedia über Sixel (englisch) `https://en.wikipedia.org/wiki/Sixel`
- All About SIXELs (Chris F. Chiesa, englisch) `https://www.digiater.nl/openvms/decus/vax90b1/krypton-nasa/all-about-sixels.text`
- LibSixel — eine C–Library für Sixel Ausgabe mit vielen Informationen rund um Sixel `https://github.com/saitoha/libsixel`

Links

4th — `https://thebeez.home.xs4all.nl/4tH/`

Der Forth–Quellcode für die Sixel — `https://github.com/cstrotm/forth-sixel`

DX–Forth auf FreeDOS mit ANSI–Escape–Sequenzen

Fred Behringer

In comp.lang.forth beklagt sich ein Teilnehmer [2] darüber, dass Forth i.A. keine Mittel bereitstellt, mit denen man ansi.sys innerhalb von Forth einsetzen kann. Anton Ertl [2] entgegnet, dass das ja nicht nötig sei, da die üblichen Forth–Systeme (z.B. auch GForth) genügend Bausteine enthielten, die das überflüssig machten. Ich beschreibe hier, wie ich vorgegangen bin, um ansi.sys innerhalb von DX–Forth, einem sehr vollständigen ANS–Forth, einsetzen zu können. Genauer: nansi.sys, da ich DX–Forth auf FreeDOS laufen lasse. (FreeDOS räumt mit den Unzulänglichkeiten von FAT16 auf). DX hätte über dos–io eigentlich schon den Zugang zu nansi.sys . Wie aber unter der virtuellen Maschinerie namens Forth nicht anders zu erwarten, wirklich nur eigentlich. Ich zeige hier, wie ich vorgegangen bin, um die Anbindung an das DOS–basierte nansi.sys zu bekommen. Der Artikel dient mir zum Brainstorming. Vollständigkeit der Liste der aufgeführten Escape–Sequenzen ist nicht mein oberstes Ziel.

Die Vorbereitung für dieses Heft 2 des Jahrgangs hat auch ihr Gutes:

Soeben erfahre ich von Carsten Strotmanns Vorhaben, in der VD (in loser Folge) über Escape–Sequenzen zu schreiben [7]. Das gibt mir den Mut, mein eigenes Vorhaben (den vorliegenden Artikel) vorzeitig zu beenden und Dinge, die mir noch nicht ganz geheuer erscheinen, vorläufig offen zu lassen — in Erwartung dessen, was von Carsten zu lesen sein wird.

In DX–Forth (siehe [1]) gilt:

`BIOS-IO ( - )` Set console output and keyboard input to use BIOS calls. BIOS–IO is the default mode. It provides color and window support without need for ANSI.SYS. (See: DOS–IO)

`DOS-IO ( - )` Set console output and keyboard input to use DOS calls. May be used to support commandline redirection, screen pausing and control–C/Break in applications.

Note: Color and windowing functions do not function in DOS–IO mode. Control–C/Break keys are not trapped and will cause an immediate exit to DOS. (See: BIOS–IO)

In Turbo–Forth und ZF reicht ansi.sys

In diesen beiden Systemen[1]funktioniert ansi.sys ohne Weiteres. Auch wenn ich sie unter FreeDOS[2] verwende.

Zur Einstimmung gebe ich zwei Beispiele an, eins für die Cursor–Steuerung und eins für die Veränderung von Farben.

Beispiel 1

Die ANSI–Escape–Sequenz

```
27 emit 91 emit
50 emit 74 emit
```

setzt den Cursor in die linke obere Ecke, löscht den gesamten Bildschirm und setzt den Forth–Prompt `OK` an den Anfang der Zeile drunter.[3]

Beispiel 2

Die ANSI–Escape–Sequenz

```
27 emit 91 emit
ascii 4 emit
ascii 5 emit
ascii m emit
```

verleiht dem Hintergrund des Forth–Prompts `OK` die Farbe Lila.

—

Der Vollständigkeit halber will ich erwähnen, dass ich auch mit ansi.com [5] sehr gute Erfahrungen gemacht habe. Da geht alles bestens. Sei es unter FAT16 oder auch unter FAT32 (wie bei FreeDOS). Und mitten im Betrieb kann man ansi.com auch aus- und wieder einschalten.

Im weiteren Verlauf gehe ich jetzt aber zu DX–Forth unter FreeDOS mit nansi.sys über.

Drei weitere Beispiele

Sie werden alle von der DOS–Eingabe–Aufforderung — dem PROMPT — eingeleitet, noch bevor Forth aufgerufen wurde. Ich werde diese Prompt–bezogenen Dinge dann aber hier nicht mehr weiter verfolgen.

Das erste Beispiel erzeugt einen Standard–Prompt `$n$g` (inverse Darstellung ausgenommen). Das zweite Beispiel stellt auf farbigem Bildschirm den PROMPT von DOS so ein, dass er das Verzeichnis `$p$g` anzeigt, und dass er das in den Farben rot gegen blau tut. Das dritte Beispiel zeigt auf dem Bildschirm — dem DOS–Bildschirm — in der rechten oberen Ecke die Zeit an, wenn man den

[1] Turbo–Forth ist das von mir bevorzugte System.

[2] FAT32–basiert

[3] Wird in manchen Forth–Systemen PAGE genannt.

DOS–PROMPT zuvor in der autoexec.bat (von DOS) untergebracht hat:

```
PROMPT $e[7m$n$p$e[m
PROMPT $p$g$e[33;44m
PROMPT $e[s$e[1;69H$t$h$h$h$e$u$p$g
```

In DX–Forth muss man ...

das System auf dos-io geschaltet haben. Normalerweise, gleich nach Aufruf von DX, steht das System auf bios–io (nicht auf dos–io). Das gilt auch im Falle, dass ein Fehler aufgetreten ist. Ich kann andererseits aus dieser Möglichkeit der Umschaltung für meine Entwicklungen Nutzen ziehen: Für den Einsatz von ansi.sys muss dos–io eingeschaltet sein. Ich kann aber auch mitten im Betrieb bios–io aufrufen und erhalte dann die ANSI–Escape–Sequenz nicht in ihrer Wirkung, sondern einfach nur eben diese Escape–Sequenz selbst auf dem Bildschirm, ASCII–Zeichen für ASCII–Zeichen, dargestellt.

Die Direkt–Eingaben sind zu umständlich

Die Frage ist, was davon forth–gerecht automatisiert werden kann. Dazu darf ich schnell ganz allgemein die Syntax der Escape–Sequenzen besprechen. Die Aufbereitung zur Verwendung nach Aufruf des Forth–Systems erledige ich dann im unten stehenden Listing.

Escape–Sequenzen–Syntax

ANSI–Escape–Sequenzen haben immer dieselbe Syntax. Mehrere Einzel–Sequenzen können, wenn überhaupt, dann aber ohne Zwischenraum (!), aneinandergereiht werden. Deren allgemeine Form lautet:

ESC [PARAMETER BEFEHL

Zur besseren Lesbarkeit habe ich hier die einzelnen Bestandteile durch je einen Zwischenraum auseinandergezogen. Dabei gilt:

`ESC` ist die Taste mit dem ASCII–Code 27d. *`ESC[`* ist das 2–Byte–Zeichenpaar, mit welchem — bei Direkteingabe vom aufgerufenen Forth–System aus — die Ausführung der zunächst noch unbekannten Escape–Sequenz eingeleitet werden soll. Man erreicht es, also das Paar[4], wenn man (z.B. über die Tastatur) `ascii 27 emit ascii 91 emit` eingibt. In DX–Forth und auch in anderen Nicht–F83–Systemen gibt es das Forth–Wort `ascii` nicht. Dort und in anderen ANS–Forth–Systemen übernimmt das system–smarte Paar `char` und `[char]` dessen Rolle. In DX–Forth, und im weiteren Verlauf meines Berichts, ist also bei Direkt–Eingabe zur Erlangung von `ESC[` einzugeben: `char 27 emit char 91 emit` . Das gilt aber nur für die Direkt–Eingabe! In einer Colon–Definition muss

```
: xxx [char] 27 emit [char] 91 emit ;
```

stehen. Nichtbeachtung führt zur Fehlermeldung und zum Ausstieg auf die DOS–Ebene.

[4] In Turbo–Forth und in ZF.

Auf der DOS–Ebene gibt es den Befehl PROMPT, über den z.B. die Uhrzeit mit Hilfe einer ANSI–Escape–Sequenz eingestellt werden kann. Im Zusammenspiel mit diesem DOS–PROMPT übernimmt die Zeichenfolge `$e[` die Rolle von `ESC[` . Ich interessiere mich im vorliegenden Artikel nur dafür, Escape–Sequenzen so aufzubereiten, dass sie innerhalb des schon aufgerufenen Forth–Systems verwendet werden können. Ich darf also im weiteren Verlauf alles, was mit dem PROMPT von DOS und speziell mit `$` zusammenhängt, ignorieren.

PARAMETER legt bei einer ganzen Gruppe von Escape–Befehlen die bestimmende Eigenschaft, z.B. die Farbe der Farbeinstellung, fest. Aber Achtung: Die Farbe Lila beispielsweise hat den Escape–Wert 45 (siehe unten), muss aber Ziffer für Ziffer, also als `char 4 emit char 5 emit` (bei Direkt–Eingabe), oder als `[char] 4 emit [char] 5 emit` (innerhalb einer Colon–Definition) eingegeben werden.

BEFEHL ist schließlich der Ein–Byte–Befehl, der die betreffende Sequenz (z.B. `ascii m` für Farb–Einstellung) zur Ausführung bringt.

Die ganz oben gebrachten 2 Beispiele

sehen unter DX–Forth also bei der Eingabe in einer Colon–Definition so aus:

```
DECIMAL

: ED ( -- )
  27 emit 91 emit
  50 emit 74 emit ;

: LILA-OK ( -- )
  27 emit 91 emit
  [char] 4 emit
  [char] 5 emit
  [char] m emit ;
```

Hierbei ist `50` der ASCII–Code der Ziffer 2 und `74` der des Großbuchstabens J.

Voraussetzung

Ich gehe davon aus, dass ich mir DX–Forth besorgt habe [1]. Weiter gehe ich davon aus, dass ich mir FreeDOS besorgt habe — kriegt man z.B. über SARDU. Und zudem gehe ich davon aus, dass ich in der config.sys–Datei von FreeDOS die Zeile `device=nansi.sys` eingebaut habe.

Zur Anbindung an DX–Forth kann nach Aufruf von DX das unten stehende Listing verwendet werden, indem man eingibt:

```
include listing.txt
```

Mit diesem Listing ist man in der Lage, die von nansi.sys bereitgestellten Bildschirm- und Cursor–Steuerbefehle nicht nur auf DOS–Ebene, sondern auch innerhalb von DX–Forth zu verwenden. Ich liste im Folgenden die in [4] gebrachten ANSI–Escape–Sequenzen auf (zumindest eine

Auswahl daraus) und verwende die dortigen (englischen) Kurzbezeichnungen der Steuerbefehle zum Aufbau meiner escape–getriebenen Forth–Worte. Eine Übertragung auf andere Forth–Systeme erscheint mir durchaus möglich.

ANSI.SYS Escape Sequences [4]

Ich zitiere und lasse alles, was mit dem DOS–Prompt zu tun hat, weg. Sobald man das Forth–System aufgerufen hat, hat der DOS–Prompt wenig Sinn. Man muss ja dann zu dessen Verwendung sowieso, über bye, auf den DOS–Unterbau von Forth zurückgehen und hat dort das ursprüngliche nansi.sys, also ohne meinen Ergänzungs–Überbau des unten stehenden Listings, zur Verfügung.

Achtung: Bei den funktionsbestimmenden Ein–Byte–Befehlen (das jeweils letzte Zeichen der Einzel–Escape–Sequenz) muss man zwischen Groß- und Kleinschreibung unterscheiden.

Beispielhaft hier nun die Ansi–Cursor–Steuer–Funktionen:

ESC[#;#H - Cursor Position (CUP). Das erste # legt die Zeilennummer fest, das zweite # die Spalte. Default ist in beiden Fällen 1.

ESC[#A Cursor Up (CUU). Schiebt den Cursor # Zeilen nach OBEN.

ESC[#B Cursor Down (CUD). Schiebt den Cursor # Zeilen nach UNTEN.

ESC[#C Cursor Forward (CUF). Schiebt den Cursor # Spalten nach RECHTS.

ESC[#D Cursor Backward (CUB). Schiebt den Cursor # Spalten nach LINKS.

ESC[#;#f Horizontal & Vertical Position. Dasselbe wie ESC[#;#H)

ESC[s Save Cursor Position (SCP). Speichert die momentane Cursor-Position. Sie kann über Esc[u wiederhergestellt werden.

ESC[u Restore Cursor Position (RCP). Stellt die über die (SCP)-Sequenz ESC[s gespeicherte Cursor–Position wieder her.

ESC[2J Erase Display (ED). Löscht den Bildschirm und setzt den Cursor in die obere linke Ecke.

ESC[K Erase Line (EL). Löscht die momentane Zeile ab der Cursor-Position.

ESC[#;...;#m Set Graphics Rendition (SGR). An die Stelle ... kann beliebiger Text eingefügt werden. Die Escape–Parameter # für SGR sind:

0 Alle Attribute AUS

1 Fettschrift AN

4 Unterstrichen AN (nur monochrom)

5 Blinken AN

7 Inverses Bild AN

8 Verbergen AN

30 Schwarzer Vordergrund

31 Rot

32 Grün

33 Gelb

34 Blau

35 Magenta

36 Cyan

37 Weiß

40 Schwarzer Hintergrund

41 Rot

42 Grün

43 Gelb

44 Blau

45 Magenta

46 Cyan

47 Weiß

ESC[=#h Set–Modus (SM)

Escape–Parameter für Set–Modus SM

0 40x25 schwarz/weiß

1 40x25 Farbe

2 80x25 schwarz/weiß

3 80x25 Farbe

4 320x200 Farbe

5 320x200 schwarz/weiß

6 640x200 schwarz/weiß

7 Bei Zeilenende auf neuer Zeile weiter: EIN

ESC[=#I Reset–Modus (RM). Verwendet dieselben Parameter wie der Set–Modus (SM)

Die Tastatur–Umbelegungen darf ich auch fortlassen. Ich hebe sie mir für später auf. Hier noch ein paar schnelle Beispiele für Escape–Sequenzen. Genaueres im Listing.

ESC[2J Löscht Bildschirm und setzt Cursor in Ecke oben links.

ESC[7m Alle nachfolgenden Zeichen in inverser Darstellung.

ESC[7;5m Alle nachfolgenden Zeichen invers und blinkend.

ESC[13;40f Platziert den Cursor in die Bildschirm–Mitte.

Experimente mit ansi.com

Unter DX–Forth ist es nicht ganz sicher, ob der nansi.sys–Mechanismus immer richtig arbeitet. Das ist es ja gerade, was ich hier herauskriegen möchte. Ich habe aber immer noch die Möglichkeit, ansi.com von Michael Mefford [5] einzusetzen. Das wirkt wahre Wunder. Es kann mitten im Betrieb ein- und auch wieder ausgeschaltet werden. Zwar nicht ganz mitten aus DX heraus, aber nach kurzem `bye` zur DOS–Ebene und Eingabe von `ansi/u`. Es erscheint dann die Meldung „ansi.com jetzt uninstalliert“ — oder so ähnlich, daher das /u — und man hat das DX–System nach Neueingabe von DX wieder im Normalzustand. Aber Achtung: Zunächst wieder bios-io, nicht dos-io, zunächst also kein ansi.com, dos-io muss erst wieder eingeschaltet werden.

Literatur und Web–Links

[1] DX–Forth:
Kann man sich über Taygeta besorgen.

[2] Ertl, M. Anton:
`www.complang.tuwien.ac.at/anton/home.html`

`comp.lang.forth` FAQs:
`http://www.complang.tuwien.ac.at/forth/faq/toc.html`

New standard:
`http://www.forth200x.org/forth200x.html`

[3] FreeDOS:
Bekommt man in beliebigem Umfang über FTP von Taygeta.Com

[4] Harris, P., Michael:
ANSI.SYS Escape Sequences. CIS 415L.
viking.delmar.edu

[5] Mefford, Michael, J.:
ANSI.COM . Aus dem PC Magazine des Jahres 1988.

[6] Schulz, Hajo:
Wie damals: ANSI–Escape–Sequenzen in der Windows–Eingabeaufforderung. c't–Heft 19/2016.

[7] Strotmann, Carsten:
Beginn einer VD–Artikelfolge über Escape–Sequenzen.

[8] Turbo–Forth:
Bekommt man ebenfalls über den FTP–Server `taygeta.com` .

Listing

```
\ Zum DX-Forth auf FreeDOS - ANSI-Escape-Sequenzen
\ Fred Behringer

\ Achtung: Hier geht es um Einstellungen, die INNERHALB
\ von Forth ueber nansi.sys am zugrundeliegenden
\ FreeDOS vorgenommen werden. Es wird also nicht
\ ueberraschen, wenn nach Verlassen von Forth
\ Veraenderungen, die man unter Forth zurueckzustellen
\ vergessen hat, im zurueckgelassenen DOS weiterhin
\ bestehen bleiben.
\ Abhilfe: nansi.sys in FreeDOS pur einschalten,
\ also nicht bei laufendem Forth.

\ Das vorliegende Listing sollte nach aufgerufenem
\ DX-Forth (DX [ret]) ueber die Direkteingabe von
\  include listing.txt
\ (wenn "listing.txt" der Name ist, den man der
\ einzuladenden Datei gegeben hat) in Gaenze
\ compiliert und dem DX einverleibt werden.
\ Ueber words (nach genuegend vielen Zeilen durch
\ [ret] abbrechen) kann man sich einen schnellen
\ Ueberblick ueber meine Worte zur nansi-Anbindung
\ an DX-Forth verschaffen.

\ Im Uebrigen darf ich mich an [4] anlehnen, wobei ich
\ allerdings alles weglasse, was mit dem Prompt von DOS
\ zu tun hat. Sobald man sich in DX befindet, hat der
\ Prompt von DOS wenig Sinn. Man muss ja dann zu dessen
\ Verwendung sowieso (ueber bye) auf den DOS-Untersatz
\ von DX-Forth zurueckgehen und hat dann dort das
\ urspruengliche nansi.sys (also ohne meinen Ueberbau
\ zur Anbindung an DX) zur Verfuegung.

\ Achtung:
\ Bei den funktionsbestimmenden Ein-Byte-Befehlen
\ (also bei dem jeweils letzten Zeichen der
\ Einzel-Escape-Sequenz) muss man zwischen Gross- und
\ Kleinschreibung unterscheiden.

\ Meine Anbindung von nansi.sys an DX-Forth laeuft
\ ueber esc[ . Die Parameter liegen in ueblicher
\ Dezimalform auf dem Datenstack und muessen ueber
\ par# usw. in eine escape-sequenz-gerechte Form
\ gebracht werden. Bei den funktionsbestimmenden
\ 1-Byte-Befehlen (dem letzten Zeichen der jeweiligen
\ Einzel-Escape-Sequenz) verwende ich fuer das
\ zugehoerige Forth-Wort die im Text aufgelisteten
\ (englischen) Kurzbezeichnungen.

\ Zwischen den Sequenzen habe ich Anmerkungen
\ eingefügt, zu dem, was mir so aufgefallen ist bei
\ deren Verwendung.



: esc[ 27 emit 91 emit ;
\ Die Einleitung jeder Escape-Sequenz

: par# ( n1 -- ) dup
  \ n1 verlangt: 0 <= Dezimalzahl n1 <= 99
  \ Beliebig viele fuehrende Nullen moeglich.
   dup 10 / 10 * -   \ Erste Ziffer ...
   dup -rot - 10 /        \ ... 0 ist, dann ignorieren;
  ?dup if 48 + emit then  \ sonst ausfuehren.
   48 + emit ;       \ Zweite Ziffer immer ausfuehren.
\ Wird in CUU gebraucht - siehe auch weiter unten.
\ Beispiel:
\ Cursor 3 Zeilen nach oben: 3 cuu
\ Fuehrende Null(en) koennen weggelassen werden.
\ Cursor 6 Spalten nach links: 06 cub

: asc(;) 59 emit ;
\ Das Trennzeichen

: par#;# ( n1 n2 -- )
\ 2 Dezimalwerte ausgeben.
\ Jedes # wie bei par#.
   swap     \ n1 n2 ==> n2 n1 (Reihenfolge beachten?)
   par#     \ Erster 2-stelliger Dezimalwert n1
   asc(;)   \ Parameter-Trennung
   par# ;   \ Zweiter 2-stelliger Dezimalwert n2
\ Wird in CUP gebraucht - siehe weiter unten.
\ Beispiel:
\ Cursor 10 Zeilen nach oben
\ und 20 Spalten nach rechts: 10 20 cup
\ Auch fuehrende Nullen sind moeglich: 010 20 cup

\ 3 Dezimalwerte.
: par#;#;# ( n1 n2 n3 -- )  \ Jedes # wie bei par#.
```

```
    swap rot  \ n1 n2 n3 ==> n3 n2 n1
    par#      \ Erster 2-stelliger Dezimalwert n1
    asc(;)    \ Parameter-Trennung
    par#      \ Zweiter 2-stelliger Dezimalwert n2
    asc(;)    \ Parameter-Trennung
    par# ;    \ Dritter 2-stelliger Dezimalwert n3
\ Wird in SGR gebraucht - siehe weiter unten.
\ Beispiel:
\ Zeichen blinkt, auf weissem Hintergrund,
\ das Zeichen ist rot: 5 47 31 sgr
\ Auch fuehrende Nullen in beliebiger Zahl sind
\ moeglich: 005 47 31 sgr
\ Die Reihenfolge ist (zumindest in diesem Fall) egal!
\ Die Parameter habe ich mit Redundanz versehen.
\ Will man das (blinkende) Zeichen schwarz werden
\ lassen, aber sonst nichts veraendern, dann erreicht
\ man das mit: 30 30 30 sgr
\ Der Stack muss aber bei sgr mindestens 3 Parameter
\ enthalten. Einfacher geht es mit sgr1
\ (siehe weiter unten).
\    Weitere Bemerkungen zu Multi-Parametern
\ finden sich auch bei sgr unten!
\ Ueberhaupt sammle ich Beispiele zur generellen
\ Ueberpruefung von Fragen, die auftreten koennen,
\ ganz am Ende des Listings.

: cup ( n1 n2 -- ) esc[ par#;# [char] H emit ;
\ Hier kommt es natuerlich auf die Reihenfolge von
\ Zeile n1 und Spalte n2 an.
\ Man sollte 0<=nx<=99 dezimal beachten.

: cuu ( n1 -- ) esc[ par# [char] A emit ;

: cud ( n1 -- ) esc[ par# [char] B emit ;

: cuf ( n1 -- ) esc[ par# [char] C emit ;

: cub ( n1 -- ) esc[ par# [char] D emit ;

: cuh ( n1 n2 -- ) esc[ par#;# [char] f emit ;

: scp ( n1 n2 -- ) esc[ [char] s emit ;

: rcp ( n1 n2 -- ) esc[ [char] u emit ;
\ Es ist mir nicht klar geworden, wo bei scp die
\ Position "gesaved" wird und wo sie im Falle rsp
\ wieder abgelesen werden kann.
\ Da ist dann wohl doch Forth dagegen?

: ed esc[ [char] 2 emit [char] J emit ;
\ Nicht wundern, dass der Bildschirm nicht total
\ geloescht wird: Das OK von Forth muss ja noch in
\ Erscheinung treten und der Cursor wird auf die
\ naechste Zeile platziert. "Verbergen" des Cursors
\ (ueber 8 sgr1) bringt nichts. Verborgen wird
\ (ab dann) nur das Zeichen im Vordergrund. Auch
\ das Forth-"OK" muss man noch kunstvoll beseitigen.
\ Mittel dazu gibt es in Forth genuegend.
\ Man beachte auch bsp1 und bsp3 am Ende des Artikels.

: el esc[ [char] K emit ;

: sgr esc[ par#;#;# [char] m emit ;

: sgr1 esc[ par# [char] m emit ;
\ Nur jeweils EIN Parameter.
\ Mehr Erlaeuterungen zu sgr1 weiter unten.
\ Beispiel: 1 sgr1 5 sgr1 45 sgr1 liefert Fettzeichen,
\ das blinkt und auf lila Hintergrund ruht.
\ Auf die Reihenfolge kommt es nicht an.
\    Die Escape-Parameter fuer sgr habe ich im Text
\ bereits angegeben. Sie liegen "normalerweise" auf dem
\ Stack. Es koennen aber natuerlich auch leicht weitere
\ Forth-Befehle formuliert werden, die die Parameter
\ schon in der Colon-Definition enthalten,
\ s. Bemerkungen zu Multiparametern weiter unten.
\    Das Forth-Wort 0 sgr setzt alle Attribute zurueck.
\ Wenn man das nicht beachtet, dann bleiben sie auch
\ nach exit zu DOS erhalten.
\ Abhilfe: DX-Forth aufrufen, durch
\   include listing.txt
\ ergaenzen, mit 0 sgr1 erledigen. Das Blinken EIN
\ (5 sgr1) bezieht sich nur auf das Zeichen unter
\ dem Cursor (und, wenn nicht abgeschaltet,
\ auf alle weiteren Zeichen).
\ Es bezieht sich NICHT auf den DOS-Cursor!
\ Aehnliches gilt fuer Verbergen EIN.
\ Es wird nicht etwa der DOS-Cursor verborgen.
\ Was verborgen (also unsichtbar gemacht) wird, ist
\ nur das Zeichen ab der Cursor-Position!
\    Bemerkungen zu Multiparametern bei sgr:
\ Es gibt keine Einzelrueckstellungen. Man muesste sich
\ mit 0 0 0 behelfen. Bei den drei (!) Parametern von
\ sgr koennen einzelne weggelassen werden. Das fuehrt
\ aber zu Unuebersichtlichkeiten. Ich habe es so
\ eingerichtet, dass der Forth-Befehl sgr auf jeden
\ Fall drei Parameter auf dem Stack vorfinden muss.
\ Nichtbeachtung fuehrt zum Fehler-Ausstieg, und damit
\ zum Forth-Prompt. DX wird im Fehlerfall wieder
\ zurueck zu bios-io geschaltet! Die Escape-Sequenzen
\ stehen dann nicht mehr zur Verfuegung. Eine Eingabe
\ von dos-io behebt diesen Unmuss.
\    Wenn man bei sgr nicht mehr weiterweiss: Eingabe
\ von 0 0 0 sgr stellt alle Parameter auf "Normalwerte"
\ (was auch immer das sein mag) zurueck.
\ Das ist (bei 0 0 0) aber doppelt (ja sogar dreifach)
\ gemoppelt.
\    Wenn man nur einen einzigen Parameter verstellen
\ will, kommt man mit x x x sgr
\ (also z.B. mit 45 45 45 sgr fuer lila Hintergrund)
\ hin. Aber was ist der Unterschied zwischen
\ 1 45 33 sgr und 33 1 45 sgr oder 45 33 1 sgr ?
\ Keiner! Ich brauchte also in sgr nicht auf die
\ Reihenfolge der Parameter auf dem Stack zu achten.
\    Ueberhaupt kann ich mir die Aufgabe erleichtern,
\ indem ich jede einzelne Einstellung in sgr mit
\ "n1 : sgr1 esc[ par# [char] m emit ;"
\ erledige.
\    Mit anderen Worten: Ich kann nicht erkennen, warum
\ in den Unterlagen bei sgr der Ausfuehrungsbefehl m
\ auf genau drei Parameter zu wirken hat. Man kann sich
\ ja auch sgr mit vier Parametern vorstellen
\ (z.B. 1 5 45 7). Oder eben nur jeweils einen einzigen
\  Parameter! Das ist der Grund, warum ich neben sgr
\ auch noch sgr1 bereitgestellt habe - und nur mit
\ diesem in Zukunft zu arbeiten gedenke.
\    Immer daran denken: Vorgenommene sgr-Einstellungen
\ bleiben bestehen, wenn sie nicht ueber 0 sgr1
\ weggenommen werden. 5 sgr1 eingegeben, laesst alle
\ neu eingegebenen Zeichen blinken. Das kann nur durch
\ Eingabe von 0 sgr1 unterbunden werden.
\    Aber: Zeichen auf dem Bildschirm, die vorher
\ geblinkt haben, blinken auch nach Abschaltung durch
\ 0 sgr1 weiter. Das heisst, die Abschaltung
\ (ueber 0 sgr1) wirkt nur ab Zeitpunkt und Ort der
\ Abschaltung (Der DOS-Cursor wird von dieser
\ Abschaltung nicht beeinflusst). Sie wirkt nicht auf
\ die zurueckliegenden Zeilen des Bildschirms,
\ wird aber beim Scrollen mit hochgezogen.
\    Vorsicht:
\ Bei scp und rcp verlangt die Logik der Dinge die
\ Abhaengigkeit der Parameter von der Reihenfolge.

: sm ( n1 -- )   \ 0<=n1<=8
  esc[ [char] = emit par# [char] h emit ;
\ Die Escape-Parameter fuer sm habe ich schon im Text
\ besprochen.
\ 0 sm erinnert mich an den VC-20-Bildschirm.
\ Mein DX-Bildschirm laeuft unter 2 sm .
\    Nicht ueberrascht sein, wenn es am Zeilenende
\ nicht weitergeht. Das, was eigentlich ueberrascht,
\ ist der Umstand, dass ueberhaupt (natuerlich nur
```

```
\ bei Arbeiten mit Escape-Sequenzen) das
\ Zeilen-Wrapping erst eingeschaltet werden muss
\ (also 7 sm), bevor es (was ja eigentlich
\ selbstverstaendlich sein sollte) zur Wirkung kommt.

: rm ( n1 -- )   \ 0<=n1<=8
  esc[ [char] = emit par# [char] I emit ;

\ rm verwendet dieselben Parameter wie der Set-Modus
\ sm .
\ Leider kann ich nicht erkennen, dass bei rm
\ irgendetwas zurueckgestellt wird. Da ist wohl
\ DX-Forth dagegen? Allerdings laesst sich schon ueber
\ sm das Bildschirmformat beliebig umschalten.
\ Leider (und nochmal leider) nur von 80x25 nach
\ 40x25 und umgekehrt.
\    Ein wesentlicher Bestandteil der ANSI-Escape-
\ Sequenzen sind ausserdem noch Befehle zur Umbelegung
\ der Tastatur. Doch das ist ein Kapitel fuer sich.
\ Ich darf mir das fuer spaeter aufheben.



\ Zum Abschluss des Listings jetzt noch ein paar
\ === Experimente zur Aufhellung ===

: bsp1 dos-io 0 sgr1 30 sgr1 41 sgr1 ed ;
\ ANSI-Einleitung, Graphik-Ruecksetzung,
\ Forth-OK schwarz, Bildschirm sonst total rot.
\ Bleibt rot, auch bei Scrolling mit OK.

: bsp2 dos-io 0 sgr1 30 cuu 34 sgr1 47 sgr1 ed ;
\ Den Cursor hochzuziehen, um das Bild von ihm zu
\ befreien, hat wenig Sinn! Durch 8 sgr1 (siehe bsp3)
\ wird NUR das Zeichen verborgen!
\ Forth-OK blau, Bildschirm sonst weiss.
\ Jetzt aber "Bildstoerung" durch OK (Scrolling).

: bsp3 dos-io 0 sgr1 30 sgr1 8 sgr1 ed ;
\ Sie sollten hier ein total-schwarzes Bild haben.
\ Nichts bewegt sich mehr.
\ Und jetzt!? Versuchen Sie es mit der Direkt-Eingabe
\ von bios-io. Experimentierfreude waere gut.

: bsp4 dos-io 47 sgr1 ed ;
\ Man kann von dem Schwarzbild nach Aufruf von bsp3
\ aus auch einfach (blind) das Forth-Wort bsp4
\ eingeben: Und schon ist alles in Butter.


\ ------------------------------------------------
\ Abschliessende Bemerkung: Es ist schwierig, Harmonie
\ zwischen nansi.sys und dem DX-Forth-Prompt
\ (sprich: dem Tastatur-Mechanismus von DX)
\ herzustellen.
\    Hier sehe ich weitere Betaetigungsfelder.
\ Fuer diesmal sei's aber genug.
```

```
Installed at PS/2 port

Modules using memory below 1 MB:

  Name           Total           Conventional       Upper Memory
  --------  ----------------   ----------------   ----------------
  SYSTEM      16,784   (16K)     10,480   (10K)      6,304    (6K)
  COMMAND      4,064    (4K)          0    (0K)      4,064    (4K)
  UDVD2        2,000    (2K)          0    (0K)      2,000    (2K)
  FDAPM          928    (1K)          0    (0K)        928    (1K)
  CTMOUSE      3,104    (3K)          0    (0K)      3,104    (3K)
  SHSUCDX     11,008   (11K)          0    (0K)     11,008   (11K)
  Free       722,144  (705K)    643,552  (628K)     78,592   (77K)
  Drives Assigned
Drive  Driver   Unit
  D:   FDCD0001   0
2 drive(s) available.

Done processing startup files C:\FDCONFIG.SYS and C:\AUTOEXEC.BAT

Type HELP to get support on commands and navigation.

Welcome to the FreeDOS 1.2 operating system (http://www.freedos.org)

C:\>_
```

Abbildung 1: Unter FreeDOS liefert DEVICE=NANSI.SYS in CONFIG.SYS den Zugriff auf die Escape-Sequenzen.

Drehgeber

Martin Bitter

Eine Auftragsarbeit: Mein Sohn möchte, dass ich für meine Enkelinnen den Spielherd eines großen schwedischen Möbelhauses mit Elektronikteilen ergänze. Der Herd soll vom Bedienfeld her dem 'echten' Herd in der Küche möglichst ähnlich werden. Unter anderem gibt es zwei Drehknöpfe, die endlos gedreht werden können. Das ist gut in der Küche und bei Kinderhand: ein Anschlag, der zerstört werden kann, ist erst gar nicht vorhanden.

Selbstverständlich soll das Ganze billig sein — im Idealfall gar nichts kosten.

Stichworte: Drehgeber, Encoder, inkrementell, quadratisch, Interrupt, gray code, atmega328p, amforth–6.6

Fundstücke

In einer meiner Schrottkisten findet sich noch eine Microsoft Bus–Maus, aus alten Zeiten. Die dazugehörige Steckkarte ist verloren gegangen. Diese Maus kann also ohne weitere Gewissensbisse zerlegt werden.

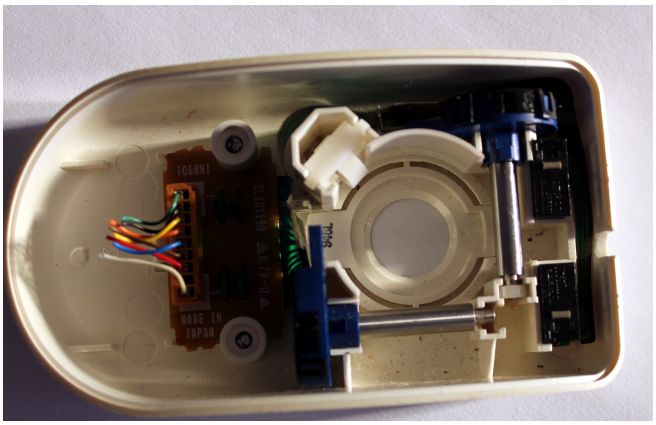


Abbildung 1: Bus–Maus geöffnet

Ihr Innenleben besteht unter anderem aus zwei elektromechnischen Drehgebern der japanischen Firma ALPS mit je drei Anschlüssen.

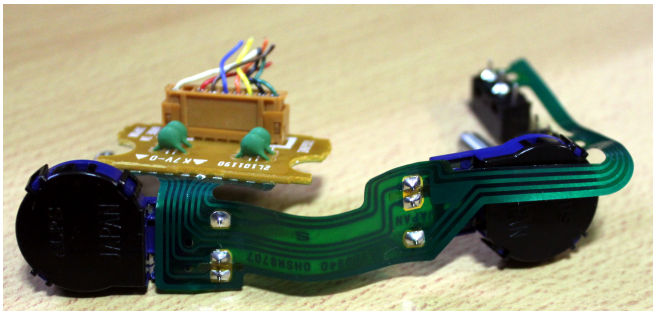


Abbildung 2: ALPS–Drehgeber

Einer dieser Anschlüsse geht an eine vergoldete Kupferfläche, die beiden anderen werden abhängig von der Stellung der Drehachse mit dieser Kupferfläche verbunden. Das führt zu festgelegten Ein–Aus–Mustern an den beiden Anschlüssen.

Die Theorie

Bei `https://www.mikrocontroller.net/articles/Drehgeber` konnte ich alles für mich Nötige über solche und ähnliche[1] inkrementellen Drehgeber erfahren: Die zwei geschalteten Anschlüsse werden Phase–A und Phase–B genannt. Sie lassen sich in vier Zustände schalten. Das Stichwort dazu ist *Quadratur*. Interpretiert man die Zustände binär, sind dies die Zustände *00 01 10 11*. Jedoch tauchen diese Zustände bei Änderungen nicht in dieser Reihenfolge auf, sondern werden als *gray code* dargestellt. D.h., zwischen zwei Schaltstellungen ändert sich jeweils nur ein Bit. Eine Änderung, die mehr als ein Bit umfasst, zeigt einen Fehler an. Die Reihenfolge der Zustände ist fest mit der Drehrichtung des Drehgebers verknüpft: *01 00 10 11 01 00 ...* steht für die Drehung im Uhrzeigersinn, *10 00 01 11 10 00 ...* für eine Drehung gegen den Uhrzeigersinn. Ein solcher Drehgeber (Encoder) zeigt nicht die absolute Stellung in Winkelgraden an, sondern ob ein Schritt in eine oder die andere Richtung erfolgt.

In dem gleichen Artikel im Mikrocontroller.net ist ein C–Code für die Abfrage eines Drehgebers wiedergegeben. Der war schnell in high–level–Forth umgesetzt. Der Algorithmus funktioniert so, dass eine Messung gespeichert wird und die jeweils aktuelle Messung mit der vorherigen verglichen wird. Durch die Feinheiten des *gray codes* (hier lohnt es sich durchaus, Gehirnschmalz in das Verständnis der Funktionsweise zu stecken) können Fehler durch Fehlabtastung (2–Bit Änderung) und Prellen erkannt werden.

Betrachten wir Prellen etwas genauer: Der letzte eingelesene Abtastwert sei *11*. Dem dürfen nun zwei mögliche Werte folgen. Entweder *10* (Gegenuhrzeigersinn) oder *01* (Uhrzeigersinn). *00* wäre mit Sicherheit eine Fehlablesung. Wir bleiben bei Abtastwerten im Uhrzeigersinn. Der abgelesene gültige Wert ist *01*. Er wird gespeichert. Prellt nun der Drehgeber, so springt der gelesene Wert zurück zu *00*, wieder vor zu *01*, wieder zurück zu *00*, wieder vor zu *01* — oder auch nicht. Betrachtet man dies jeweils zusammen mit dem zuvor abgelesenen Wert, ergibt sich in diesem Beispiel *11–10, 10–11, 11–10, ... 10–11*. Das Prellen wird auf jeden Fall mit *10–11* verlassen. Eine gängige Methode des Softwareentprellung ist das Abwarten, bis das Prellen mit Sicherheit vorbei ist,

[1] ähnliche: Das bedeutet hoffentlich, der zu entwickelnde Code trifft auf viele handelsübliche Drehgeber zu.

um dann erneut einzulesen. Hier ist das nicht nötig. Denn bei jedem Rückwärtsschritt des Prellens erniedrigen wir unseren Zähler, bei jedem Vorwärtsschritt des Prellens erhöhen wir ihn. Die Summe stimmt!

Leidige `IF-ELSE-THEN`–Abfragen werden durch eine kleine Sprungtabelle vermieden. Das spart Rechenzeit. Die Tabelle ist nicht besonders lang, es werden 16 Einträge zu 4 Bit benötigt! Je 2 Bit für den „letzten“ Wert, je 2 Bit für den aktuellen Wert. An die gültigen Positionen werden die gewünschten Zähler `+1` bzw. `-1` gesetzt, die ungültigen Positionen erhalten eine Null `0`.

High–Level–Code

In amforth–6.6 ist das in High–Level schnell implementiert: Um ein Gefühl für die Sache zu bekommen, zuerst in *plain forth* in einer Endlosschleife. Die Verwendung von `marker -enc-` erlaubt beim Entwickeln häufiges Neuladen.

```
\ drehgeber.frt
\ use an very old APS rotary device found
\ in an Microsoft Bus--Mouse
\ amforth version: 6.6
\ MCU: atmega328p
\ Board: Arduino Duemilanove
\ Phases should be 00 10 11 01 ( 0 2 3 1 )
\ or vice versa 01 11 10 00 ( 1 3 2 0 )
\
\ --enc--High--level Interrupt

marker --enc--
```

Die Worte aus dem Paket *bitnames.frt* erlauben mir eine einfache Definition der verwendeten Portpins. Ich habe hier die Port–Pins D2 und D3 gewählt, das hat Gründe, die später zur Sprache kommen. Möglich sind alle Port–Pins, die sich als Eingang schalten lassen. Mit `bitmask:` und `pin@` greife ich auf mehrere Pins gleichzeitig zu. Bei anderen Portpins muss an zwei Stellen im Code etwas angepasst werden: Zum einen die Definitionen mit `portpin:` zum anderen der Parameter fürs shiften im Wort `enc#1@`. Die Anpassung der Bitmaske erfolgt bei Kompilieren.

Bei der Intialisierung werden die Pins mit einem internen Pull–Up Widerstand beschaltet, der das *Floaten* des Signalpegels verhindert.

Der ursprüngliche Code ist für die Abfrage zweier Drehgeber geschrieben. Daher die Banamsung mit `xxx#1`. Wegen der Übersichtlichkeit, ist hier nur der Code für einen Drehgeber gezeigt.

Hauptwort ist `enc#1`, das die Signale der beiden Phasen abliest und passend für die Sprungtabelle aufbereitet.

Da bei meiner Konfiguration die zusammengehörigen Pins nebeneinander liegen, reicht eine simple Bitschieberei mittels `2 rshift` , um aus den Bitmustern des Ports `0000xx00` passende Werte `xx` zu generieren.

```
\ Arduino board pin 2
PORTD 2 portpin: phase-A
\ Arduino board pin 3
PORTD 3 portpin: phase-B

: init-rotary-ports ( -- )
  phase-A 2dup pin_input pin_pullup_on
  phase-B 2dup pin_input pin_pullup_on ;

  PORTD phase-A drop phase-B drop +
  bitmask: quadrat

: enc#1@ ( -- n )
  quadrat pin@ 2 rshift %11 and ;
```

Das Anlegen der Sprungtabelle im Flashspeicher macht das ganze resetfest. Dazu greife ich mit dem Wort `,i` direkt auf den Dictionarypointer `dp` zu. Natürlich kann man das Füllen der Tabelle mit in die Initroutine schreiben, die ja sowieso zum Setzen der Portpins aufgerufen werden muss, aber ich habe mich hier so entschieden. Ebenso habe ich mich für die halbe Auflösung entschieden.[2] Wer mag, kommentiert das aus und verwendet statt dessen die höher auflösende Tabelle.

Die Variable `enc#1` wird entsprechend den Drehungen am Drehgeber hoch– oder heruntergezählt.

Damit ist nun alles fertig für eine Abfrage des Drehgebers,

```
: ,i ( n -- )
  dp !i dp 1+ to dp ;

  Create gray_table

\ mit wackligem Rasterpunkt halbe Auflösung
  0 ,i 0 ,i -1 ,i 0 ,i
  0 ,i 0 ,i 0 ,i 1 ,i
  1 ,i 0 ,i 0 ,i 0 ,i
  0 ,i -1 ,i 0 ,i 0 ,i

\ volle Auflösung
\ 0 ,i 1 ,i -1 ,i 0 ,i
\ -1 ,i 0 ,i 0 ,i 1 ,i
\ 1 ,i 0 ,i 0 ,i -1 ,i
\ 0 ,i -1 ,i 1 ,i 0 ,i

Variable enc#1 0 enc#1 !
```

Als Demo sei hier die Abfrage des Drehgebers in einer Endlosschleife gezeigt (heavily commented). Bei Tests mit der Hand zeigten sich meist plausible Werte, im 'Prüfstand' erwiesen sie sich als genau, wie im Abschnitt *Prüfstand* beschrieben. Die 'alten' Geberwerte werden auf dem Stack übergeben. Dann wird gemessen, welchen Wert der Drehgeber meldet und dem entsprechend in der Tabelle nachgeschaut. Der resultierende Wert wird zum Zähler addiert. (Addition von `-1` entspricht einer Subtraktion). Am Ende der Schleife wird der aktuelle Zählerstand ausgegeben. In einer ersten Version habe ich den jeweils

[2] vgl.: `https://www.mikrocontroller.net/articles/Drehgeber`

aktuellen Stand in der Schleife ausgeben lassen. Das empfiehlt sich nicht! Die Ausgabe kostet soviel Zeit, dass die Abtastung nicht mitkommt und absurde Werte liefert.

Die Wartezeit fürs Prellen von 1ms ist experimentell. Sie kann verändert oder weggelassen werden.

```
: steps ( -- n ) \ zählt die Impulse eines Quadratur--Drehgebers
0                \ Dummy 'alter' Wert 2 Bits ( 0 -- 3 )
BEGIN            \ starte Schleife
2 lshift         \ 'alten' Wert um 2 bits nach links schieben
enc#1@           \ neuen Wert holen 2 Bits ( 0-- 3 )
\ 1 ms           \ etwas Warten (prellen) --> Experimentierfeld
swap over +      \ aus altem und neuen Wert einen 4-Bit-Wert erstellen ( 0 -- 15 )
gray_table + @i  \ entsprechenden Wert auslesen ( -1, 0, 1)
enc#1 +!         \ zum Zähler addieren
key?             \ Taste gedrückt?
UNTIL            \ Falls ja: fertig
key drop         \ zur Sicherheit wg key?, Tastenwert holen und wegwerfen
drop             \ 'alten' Wert wegwerfen
enx#1 @ . ;      \ Zählerstand ausgeben
```

Prüfstand

Es fiel mir gar nicht so leicht, einen Drehgeber reproduzierbar genau zu drehen. Egal wie ich es versuchte, zwirbeln zwischen den Fingern, drehen der Hand mit Umsetzen und Neugreifen, ich war mir nie sicher, ob die gemessenen Ungenauigkeiten am Aufbau und Code oder aber an Mängeln des Händischen lagen. Selbst das Ankleben eines Zeigers half mir wenig. Letztendlich baute ich aus Legotechnik–Material einen simplen Prüfstand. Jetzt kann ich reproduzierbar mehrere Umdrehungen in eine Richtung, dann wieder entgegengesetzt machen, Zahnradzähne zählen und schauen, ob der Zähler wieder auf Null geht oder ob pro Umdrehung gleiche Werte herauskommen.

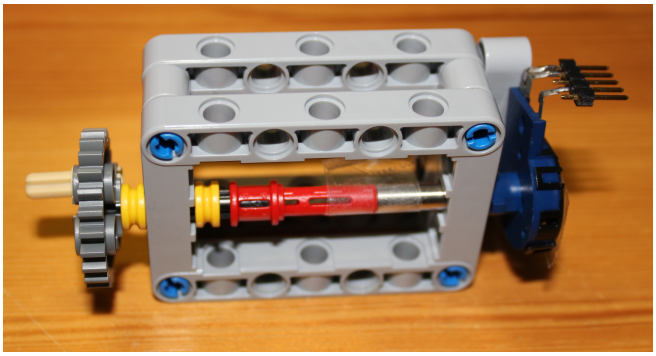


Abbildung 3: Prüfstand

Interrupt

Der Algorithmus funktioniert, aber eine Endlosschleife verbraucht (verschwendet) MCU--Zeit. Deshalb soll der Drehgeber in einem Interrupt abgefragt werden. Genauer gesagt, der Drehgeber soll einen Interrupt auslösen. Bei der Behandlung dieses Interrupts werden dann die Eingangswerte abgefragt und der Zähler entsprechend verändert. Fast alle Pins des atmega328p können als externe Interruptquellen angeschlossen werden. Generell kann ein Interrupt ausgelöst werden durch Veränderungen an einem beliebigen Pin eines Ports — dann muss die InterruptServiceRoutine (ISR) sich darum kümmern, welcher Pin genau den Interrupt ausgelöst hat und auch, ob der Interrupt bei einer fallenden oder steigenden Flanke erfolgte. Eine weitere Möglichkeit bieten die externen Interruptquellen an den Pins D2 und D3. Diese sind feiner konfigurierbar.

Es soll ein zweiter Drehgeber verwendet werden. Daher habe ich Folgendes entschieden: Drehgeber Nummer 1 bekommt die Pins D2 und D3, die über die externen Interrupts INT0 und INT1 ausgewertet werden — vgl. Abschnitt Low–Level–Interrupt — und Drehgeber Nummer 2 bekommt die Pins D4 und D5, die über den generellen externen Interrupt PCI PORTD (PinChangeInterrupt PORTD) ausgewertet werden. Da ich die anderen Pins des PCI PORTD nicht für einen Interrupt benutze, spare ich mir so die Rechenzeit für eine Auswertung, welcher Pin beim PCI PORTD angesprochen wurde, denn in meinem Fall ist es egal, welcher der beiden Pins des Drehgebers sich ändert. Bei INT0 und INT1 weiß ich es sowieso.

Amforth bietet schon seit einigen Versionen die Möglichkeit High–Level–Interrupts zu verwenden. Technisch geschieht dies so, dass die Interrupts mit einem Forthword vorbesetzt sind, das in eine eigene Tabelle springt und das dort stehende *execution token* ausführt. Leider ist diese Tabelle nicht vorbesetzt, d.h., uninitialisiert steht dort ein `$FFFF`. Sollte der Interrupt jetzt aufgerufen werden, landet man im Nirwana. Das *execution token* einer High–Level–Routine kann durch `' word interruptname int!` gesetzt werden. Die Feinheiten des Interrupts werden durch das Setzen diverser Bits im zugehörigen Register (`c!`, `c@`) eingestellt. Um einen Interrupt 'scharf' zu schalten, muss in dem *template.asm* (bei mir ist es umbenannt in *herd.asm*) die entsprechende Option `WANT_XX = 1` gesetzt werden -- siehe Kapitel Low–Level–Interrupt.

[3] Ob das wirklich schneller ist, habe ich nicht getestet. Es kann sein, dass der prinzipiell langsamere Flashzugriff dadurch kompensiert wird, dass auf den Flash mit Wortbreite zugegriffen wird. Natürlich spielt auch FCPU eine Rolle.

High–level–Interrupt

Die Abarbeitung eines Interrupts soll möglichst schnell geschehen. Daher lege ich die Sprungtabelle in den RAM.[3] Im Kopf des Programmes habe ich notiert, wo ich welche Optionen für 'mein' amforth setzen muss. Der weitere Code ist schon bekannt:

```
\ high--level--ISR.frt
\ use an very old APS rotary device found
\ in an Microsoft BUS Mouse
\ amforth version: 6.6
\ MCU: atmega328p
\ Board: Arduino Duemilanove
\ Phases should be 00 10 11 01 ( 0 2 3 1 )
\ or vice versa 01 11 10 00 ( 1 3 2 0 )
\ for ISR (interrupt service routine) see
\ amforth.pdf: 5.2. General Code Examples
\ -- Interrupt
\ Service Routines p85--p86
\ see also: ../devices/atmeag328p.asm
\ and amforth.pdf p107 WANT--Options

\ two rotary encoders (1, 2)

\ herd.asm (used by Makefile)
\ These are my WANT--Options:
\ cat herd.asm | grep --v ';' | grep WANT
\ .set WANT_PORTD = 1
\ .set WANT_EXTERNAL_INTERRUPT = 1
\ .set WANT_INTTERRUPTS = 1
\ .set WANT_IGNORECASE = 1

--int--

marker --int--

\ Arduino board pin 4 PCINT(20)
  PORTD 4 portpin: phaseA#2

\ Arduino board pin 5 PCINT(21)
  PORTD 5 portpin: phaseB#2

PORTD phaseA#2 drop phaseB#2 drop +
bitmask: enc#2msk

: init-enc#2-pins ( -- )
  phaseA#2 2dup pin_input pin_pullup_on
  phaseB#2 2dup pin_input pin_pullup_on ;

: enc#2@ ( -- n )
  enc#2msk pin@ 4 rshift ;

: ,i ( n -- )
  dp !i dp 1+ to dp ;

Create gray_table

\ mit wackligem Rasterpunkt halbe Auflösung
  0 ,i 0 ,i -1 ,i 0 ,i
  0 ,i 0 ,i 0 ,i 1 ,i
  1 ,i 0 ,i 0 ,i 0 ,i
  0 ,i -1 ,i 0 ,i 0 ,i

Variable gray-ram 32 allot

: copy-flash2ram ( fladdr ramadd n -- )
  \ n = bytes
  0 ?DO over @i
  over !
  2 + swap 1+ swap
  LOOP
  2drop ;
```

Den alten Zählerstand kann ich nicht mehr auf dem Stack halten, er wird in der Variablen `old_value` gespeichert. Die Sprungtabelle wird in den RAM kopiert. Das sind schon die einzigen Konzessionen, die ich zur Verwendung des Interrupts einhalten muss. Das Wort `enc#1step` ist dem ehemaligen Wort `steps` sehr ähnlich. Die Sequenz `-int ... +int` ist in diesem konkreten Fall (noch) nicht wirklich notwendig, ich verwende sie dennoch, um mir nichts Übles anzugewöhnen.

```
Variable enc#2
Variable old_value
\ Variable interrupts# \ Experimentierfeld

: enc#2step ( -- )
  -int
  \ 1 interrupts# +! \ Experimentierfeld
  \ 1 ms             \ Experimentierfeld
  old_value @ 2 lshift
  enc#2@             \ new value
  dup old_value ! +
  2* gray-ram + @
  enc#2 +!
  +int ;
```

Der Interrupt muss mit den richtigen Parametern versehen und eingeschaltet werden. Mittels `int!` patche ich das *execution token* der zukünftigen ISR (`enc#1step`) in die amforth–Interrupt–Tabelle. Im Pin–Change–Control–Register PCICR sorgt Bit 2 dafür, dass PORTD einen Interrupt generieren kann. Das Register PCMSK2 enthält die Pins, die den Interrupt auslösen. Für den Fall, dass ich überprüfen muss, welcher der Pins des PORTD den Interrupt ausgelöst hat, muss ich noch das Pin–Change–Interrupt–Flag–Register PCIFR abfragen. Das ist aber hier nicht notwendig. Falls ich den Interrupt stoppen will, schalte ich im PCMSK2 die betreffenden Pins aus.

```
: enc#2>isr ( -- )
  ['] enc#2step PCINT2Addr int! ;

: restore--interrupt--enc#2
  -1 PCINT2Addr int! ;

: init-enc#2-isr ( -- )
  \ pin change interrupt 2 i.e. PORTD
  PCICR c@ %100 or PCICR c!
```

```
  \ pins 4, 5
  PCMSK2 c@ %110000 or PCMSK2 c! ;

: stop-PCI-enc#2 ( -- )
  PCMSK2 dup c@ %11001111 and c! ;
```

Nach der Ausführung der Initialisierung `init-encoder` läuft — ohne weiteres Zutun — der Interrupt und die Schritte des Drehgebers werden gezählt. Die Anzeige geschieht mittels `enc#1 @ .`

```
: init-encoder ( -- )
  gray_table gray-ram $10 copy-flash2ram
  init--enc#2-pins
  init--enc#2-isr 10 ms
  enc#2>isr ;
```

Experimentierfeld: Zeit totschlagen

Im Quellcode gibt es drei auskommentierte Zeilen: *Experimentierfeld*. Dort kann man zählen, wie oft der Interrupt aufgerufen wurde. Das kann verglichen werden mit der Anzahl der gezählten Schritte. Wir erinnern uns: Jedes Prellen löst einen Interrupt aus, prellende Schritte heben sich auf (`+1 + -1 = 0`). Das Verhältnis der gezählten Interrupts zu den gezählten Schritten liefert einen Wert, der etwas über die Qualität von Hard– und Software–Entprellung aussagt. Als Faustregel gilt: Ist Interruptanzahl höher als die Anzahl der gezählten Schritte, ist alles OK! Cave at![4] Hier kann man aus Neugier viel Zeit verbringen.

Wenn's noch schneller gehen muss: low–level–Interrupt

Erich Wälde zeigt im Cookbook zu amforth (Doku im Ordner *../doc/amforth.pdf*), wie er einen Interrupt low–level eingebunden hat. Das heißt, ohne den in amforth eingebauten Mechanismus. Wenn es richtig schnell gehen muss (Interrupts sollen schnell sein!), dann kann man eine low–level–ISR in Assembler schreiben. Ich habe dazu eine Datei *encoder.asm* geschrieben, die im Ordner *../words/* gespeichert wird. Um sie beim Kompilieren einzubinden, muss sie dem Kompiler bekannt gemacht werden. Dies geschieht durch Eintrag der Zeile `.include "words/encoder.asm"` in der Datei *dict_ appl.inc*.

Zu Beginn wird im data–space (RAM) Platz für einige Variablen reserviert und im code–space (FLASH) die Tabelle angelegt. Je nachdem, ob später nur ein Interrupt oder beide verwendet werden, wird die Tabelle angepasst. Bei einem Interrupt habe ich eine bessere Genauigkeit, bei zwei Interrupts eine höhere Auflösung.[5]

```
; 2018-03-21 words/encoder.asm
; INT0Addr INT1Addr PIND

.set encoder1 = 0b001100 ; PORTD 2 3

; - RAM space for encoders #1
.dseg
 old_enc1: .byte 1
 enc1: .byte 2
 enc_delay: .byte 2

.cseg
enc_table:
;; halbe Auflösung
   .dw 0x0000, 0xff00, 0x0000, 0x0001
   .dw 0x0100, 0x0000, 0x00ff, 0x0000
;; volle Auflösung
;; .dw 0x0001, 0xff00, 0xff00, 0x0001
;; .dw 0x0100, 0x00ff, 0x00ff, 0x0100
```

Diesmal verbinde ich den Drehgeber mit den Pins D2 und D3, zu denen die Interrupts INT0 und INT1 gehören. An die Adressen der Interrupts werden Sprungbefehle zur jeweiligen ISR geschrieben. Beide Interrupts springen zur selben Routine. Die Interrupts INT0 und INT1 können auf diverse Auslösebedingungen eingestellt werden. Steigende oder fallende Flanke oder low–level oder jede Änderung des Pins lösen den Interrupt aus. Damit kann man spielen.

```
; register isr for INT0 INT1
.set pc_ = pc
.org INT0Addr
 jmp encoder1_count_isr
.org INT1Addr
 jmp encoder1_count_isr
.org pc_
```

Einige Funktionen und Makros machen den Code (hoffentlich!) lesbarer, der ursprünglich für zwei Drehgeber geschrieben wurde, hier der Lesbarkeit halber aber auf einen Drehgeber beschränkt wird.

Drei Makros werden definiert: `save_state` rettet die Register, die ich benutzen werde. Das Macro `restore_state` versetzt sie wieder in den ursprünglichen Zustand. Das Macro `shift_old` verschiebt den Wert im temp0–Register um zwei Stellen nach links.

```
; - isr

.macro save_state
 push xl
 in xl,SREG
 push xl
 push xh
 push zl
 push zh
 push temp0
 push temp1
 push temp2
.endmacro

.macro restore_state
 pop temp2
```

[4] lat.: Aber Vorsicht!

[5] Mir ist es nicht gelungen, bei voller Auflösung zuverlässige Werte zu bekommen. `enc_delay` bekommt hier seltsamerweise(?) einen großen Einfluss.

```
 pop temp1
 pop temp0
 pop zh
 pop zl
 pop xh
 pop xl
 out SREG,xl
 pop xl
.endmacro

.macro shift_old
 lsl temp0
 lsl temp0
.endmacro
```

Die Funktion `wait_enc` entnimmt aus der Variablen `enc_delay` einen Wert und zählt ihn herunter. Die Funktion `dispatch_table` bekommt im temp0–Register den Offset in die Sprungtabelle übergeben und liest den dortigen Wert in das X–Register ein. Eine Besonderheit hierbei ist das Umrechnen der Adressierung von `enc_table` von Word– auf Bytezugriff. Dies geschieht durch das shiften um 1 nach links.

```
wait_enc:
 ;; wait a little bit, param is in enc_delay
 ldi zl, LOW(enc_delay)
 ldi zh, HIGH(enc_delay)
 ld xl, z+
 ld xh, z
 wait_1:
 sbiw xl,1 ;
 brne wait_1 ;
 ret

dispatch_table:
 ldi zl, low(enc_table<<1)
 ldi zh, high(enc_table<<1)
 add zl, temp0
 adc zh, zeroh
 clr xh
 lpm xl, z
 sbrc xl,7
 ldi xh, $FF
 ret
```

Nun ist alles bereit, um die Hauptroutine zu gestalten: Register retten, alten und neuen Gray–Code holen, Wert aus der Sprungtabelle lesen, Wert zum Zähler addieren, Register restaurieren. Fertig!

```
encoder1_count_isr:
 save_state
 cli
 ;; combine old and new graycode
 clc ; clear carry
 lds temp0, old_enc1
 shift_old
 rcall wait_enc

 in temp1, PIND
 ldi temp2, encoder1 ; $0C ; %1100
 and temp1, temp2
 lsr temp1
 lsr temp1
 sts old_enc1,temp1
 add temp0, temp1

 rcall dispatch_table

 ;; add-sub counter
 clc ; clear carry, jibc
 lds zl, (enc1+0)
 lds zh, (enc1+1)
 add zl, xl
 adc zh, xh
 sts (enc1+0), zl
 sts (enc1+1), zh

 restore_state
 reti
```

Wirklich fertig? Nicht ganz. Dieser schöne Assembler–Code muss noch dem Forthsystem bekannt gemacht werden. Insgesamt werden es zwei *low–level–words*: eine Variable `enc#1`, die den aktuellen Zählerwert hält, und eine Variable `enc_delay`, in der der Zeitwert zum Entprellen gehalten wird. Eventuell kann nach einigem Testen `enc_delay` headerless compiliert werden.

```
; - forth variables

VE_ENC1:
 .dw $ff05
 .db "enc#1"
 .dw VE_HEAD
 .set VE_HEAD = VE_ENC1
XT_ENC1:
 .dw PFA_DOVARIABLE
PFA_ENC1:
 .dw enc1

VE_ENC_DELAY:
 .dw $ff09
 .db "enc_delay"
 .dw VE_HEAD
 .set VE_HEAD = VE_ENC_DELAY
XT_ENC_TABLE:
 .dw PFA_DOVARIABLE
PFA_ENC_TABLE:
 .dw enc_delay
```

Nach einer Neukompilation und dem Flashen von Amforth zeigt uns der Befehl `words` nun zwei neue Einträge: `enc#1` und `enc_delay`.

Es bleibt nur noch wenig zu tun: Die beiden Portpins müssen als Input mit Pull–Up–Widerständen konfiguriert werden,[6] und der Interrupt durch das Beschreiben der beiden Register EICRA (External Interrupt Configuration Register A) und EIMSK (External Interrupt Mask

[6] Es gibt Module mit Drehgebern, die eigene Pull–Ups mitbringen, dann kann auf die internen Pull–Ups des Atmega verzichtet werden.

Register) eingeschaltet werden. Ab jetzt wird jede Bewegung am Drehgeber gezählt. Mit den üblichen *words* `@`, `!` kann auf den Zähler und den Delaywert zugegriffen werden. In `register-enc-isr` entscheide ich mittels *%11* bzw *%01* oder *%10*, welcher Pin einen Interrupt auslöst. Im Falle von *%10* – *%01* hat das Auswirkungen auf die Drehrichtung, bei *%11* (beide Interrupts aktiv) muss ich die Tabelle für die volle Auflösung verwenden. Nicht passende Paarungen führen zu unsinnigen oder gar keinen Werten.

```
\ low-level-ISR.frt
\ Initialise low-level-ISR
\ Drehgeber an PIN D2 und PIN D3
\ atmega328p 16 FCPU
\ amforth-6.6

-int-

marker -int-

\ Arduino board pin 2 int0
  PORTD 2 portpin: phaseA1
\ Arduino board pin 3 int1
  PORTD 3 Portpin: phaseB1

: init-pins ( -- )
  phaseA1 2dup pin_input pin_pullup_off \ on
  phaseB1 2dup pin_input pin_pullup_off \ on ;

: register-enc-isr ( -- )
  %0101 EICRA c! \ any edge
  %01 EIMSK c! \ int0 int1 active ;

: init-encoders ( -- )
  1000 enc_delay !
  init-pins
  register-enc-isr ;

: test ( -- )
  BEGIN
  cr enc#1 @ 8 .r
  key?
  UNTIL ;
```

Darf's etwas mehr sein?

Im realen Programm werden zwei Drehgeber abgefragt. In amforth oder Assembler geschieht die Abfrage auf die Pegelstände an den dann vier Pins mit einem einzigen Lesezugriff auf PORTD. Die Auswertung geschieht getrennt nacheinander. Auf die gleiche Art können bis zu vier Drehgeber an einem Port (ein Lesezugriff) abgefragt werden. Dafür bleibt als Einziges (oder es bietet sich an) die Verwendung des Pin Change Interrupts (PCI).

Gruppenfoto der Tagung

Forth–Gesellschaft e.V. Ordentliche Mitgliederversammlung 8.4.2018

Carsten Strotmann

Moderation: Michael Kalus

Protokollant: Carsten Strotmann / Anton Ertl

Teilnehmer: Direktorium: Ewald Rieger, Ulli Hoffmann, Bernd Paysan. Insgesamt 17 stimmberechtigte Mitglieder. Anzahl Mitglieder in der Versammlung: 17

Sitzungsdatum: 08.04.2018

Sitzungsstart: 09:41 Uhr

Sitzungsende: 11:26 Uhr

Sitzungsort: Linuxhotel Essen, Antonienallee 1, 45279 Essen–Horst

Begrüßung

Ewald Rieger begrüßt im Namen des Direktoriums die anwesenden Mitglieder.

Wahl des Schriftführers

Als Protokollant wird Carsten Strotmann gewählt.

Wahl des Versammlungsleiters

Zum Sitzungsleiter wird Michael Kalus gewählt. Der Sitzungsleiter stellt fest, dass die Versammlung fristgerecht einberufen wurde. Es sind 17 stimmberechtigte Mitglieder anwesend. 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

Bericht der Verwaltung (Ewald Rieger)

Ewald Rieger hat das Forth–Büro über 20 Jahre geführt und wird die Verwaltungstätigkeit auf eigenen Wunsch zum Ende des Geschäftsjahres 2018/2019 abgeben. Es muss ein neues Forth–Büro bis spätestens zur Tagung 2019 gefunden werden. Die Anwesenden Mitglieder bedanken sich bei Ewald Rieger und seiner Frau Andrea für die hervorragende Arbeit im Forth–Büro.

Mitgliederentwicklung

Im Jahr 2017 gab es erfreulicherweise 2 Neueintritte und 4 Austritte. Das langjährige Mitglied Friedrich Prinz ist im Jahr 2017 verstorben. Der Verein hatte zum Jahresende 2017 96 Mitglieder.

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 7.866,47 Euro zum Jahresende und damit ein Vermögenszuwachs von 1.146,72 Euro.

Wie auch schon im Vorjahr wurden leider keine Mittel für Projekte und Reisekosten–Erstattung von Mitgliedern angefragt.

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. Im Geschäftsjahr 2017/2018 wurden drei Ausgaben des Magazins erstellt, PDF Versionen können nun im Wiki der Forth Gesellschaft Webseite abgerufen werden. Dank geht ausdrücklich an Michael Kalus, der vor allem Inhalte für das Heft beschafft und sich dann auch um die Herstellung kümmert. Ein Dank auch an alle Autoren, die Inhalte zum Magazin beigesteuert haben. Es wird wie jedes Jahr gewünscht, dass die Vorträge zur Tagung in Textform für die Vierte Dimension eingereicht werden.

Internet–Präsenz (Ulrich Hoffmann)

Gerald Wodni und das Webseiten–Team arbeitet weiter an dem neuen Webauftritt, die ist derzeit unter http://neu.forth-ev.de erreichbar. Viele Inhalte sind schon übertragen, es fehlt noch der Feinschliff an einigen Stellen. Aufgrund anderer Forth–Projekte konnte die neue Webseite im Jahr 2017 nicht komplett fertiggestellt werden. Arbeit an der neuen Webseite wird intensiviert mit dem Ziel, die neue Webseite noch im Jahr 2018 aktiv zu nehmen.

Außendarstellung und Projekte (Bernd Paysan)

Auch im Jahr 2017 war die Forth–Gesellschaft wieder auf Messen und Veranstaltungen präsent. Auf dem Vintage–Computing–Festival Europe (VCFe) im April 2017 in München (zum ersten Mal in neuen Räumen) sowie auf der Schwesterveranstaltung in Berlin (VCFb) wurde Forth auf alten Maschinen präsentiert und Teilnehmern sowie Besuchern die Forth–Programmierung nahe gebracht.

Im August war ein Team der Gesellschaft auf der Maker–Fair in Hannover präsent, jedoch wurde festgestellt, dass die Maker–Fair sich weg von einer Veranstaltung von

kreativen Bastel–Projekten hin zu einer Verkaufsveranstaltung entwickelt. Es ist daher nicht sicher, ob die Forth–Gesellschaft im Jahre 2018 auch wieder zur Maker–Fair nach Hannover gehen wird, oder ob stattdessen kleinere, weniger kommerzielle Veranstaltungen besucht werden.

Traditionell haben Mitglieder der Forth–Gesellschaft wieder auf dem Jahrestreffen des Chaos–Computer–Clubs, dem 34C3 Kongress, zum ersten Mal am neuen Ort in der Messe Leipzig, Forth demonstriert und fragenden Teilnehmern Rede– und Antwort gestanden.

Gründungsmitglied Klaus Schleisik gab einen Vortrag mit dem Titel "Die Programmiersprache Forth", welcher als Video unter https://media.ccc.de/search/?q=forth zu finden ist.

Bericht des Kassenprüfers (Klaus Zobawa)

Die Prüfung der Kasse fand am 07.04.2018 durch Klaus Zobawa statt. Er berichtet, dass alle Buchungen belegt und nachvollziehbar sind. Die Buchhaltung ist vorbildlich.

Entlastung des Direktoriums

Carsten Strotmann stellt den Antrag, das Direktorium zu entlasten. Der Antrag wird einstimmig angenommen, das Direktorium ist damit entlastet.

Wahl des neuen Direktoriums

Ewald Rieger kündigt an, aus persönlichen Gründen nicht mehr für das Amt eines Direktors der Forth–Gesellschaft e.V. zur Verfügung zu stehen. Bernd Paysan und Ulrich Hoffmann stellen sich wieder zur Wahl. Für das neue Direktorium schlagen die Teilnehmer der Versammlung Gerald Wodni und Carsten Strotmann vor.

Die Teilnehmer konnten je drei Kandidaten (für die drei Direktoriums–Plätze) auf die Stimmzettel schreiben.

Bei der nachfolgenden Wahl wurden 17 gültige Stimmzettel abgegeben und das folgende Ergebnis erziehlt:

Kandidaten	Stimmen
Bernd Paysan	17
Ulrich Hoffmann	16
Carsten Strotmann	14
Gerald Wodni	4

Tabelle 1: Wahlergebnis

Das neue Direktorium besteht somit aus Bernd Paysan, Ulrich Hoffmann und Carsten Strotmann. Die Gewählten nahmen die Wahl an.

Carsten Strotmann gibt an dieser Stelle die Funktion des Protokollführers an M.A. Ertl ab (per Akklamation).

Neue Projekte

MCU–Board mit Forth für Messen

Es soll ein neues Board für Messe–Demos und als einfacher Einstieg für Forth–Interessierte gefunden werden. Carsten Strotmann schlägt ein Arduino–Uno kompatibles Board vor, welches im April per Kickstarter–Kampagne finanziert wird. Das Direktorium wird verschiedene Optionen prüfen und ein Board bestellen.

Sammelbestellung RISC–V HiFive–1 Board

Matthias Koch hat Mecrisp–Forth auf die RISC–V–CPU–Architektur portiert und wünscht sich ein HiFive–1–Board mit RISC–V–CPU zum Test. Der Mikrokontrollerverleih organisiert eine Sammelbestellung, Matthias Koch bekommt ein Board für die Tests von Mecrisp. Weitere Boards können über den Mikrokontrollerverleih geliehen werden.

Neue Website

Die neue Webseite ist im Aufbau und leider noch nicht fertig geworden. Die alte Webseite wird noch für die DSGVO (neue Europäische Datenschutzverordnung) aktualisiert, ein Wechsel auf die neue Webseite soll schnellstmöglich geschehen.

Bitkanone v2

Gerald Wodni koordiniert ein Projekt zur Erstellung einer neuen Version der Bitkanone (Forth–gesteuertes Display mit RGB–LED–Streifen). Es gab auf der Tagung ein Brainstorming zum Projekt am Samstag Abend. Die neue Bitkanone soll aus einer 9x9 X–LED–Display–Kachel bestehen, mehrere Kacheln können zu einem größeren Display zusammengesteckt werden. Jörg Volkers hat angeboten, die Produktionsstraße von Tematik zur Erstellung der Bitkanone zu benutzen.

Verschiedenes

Die EuroForth 2018 findet dieses Jahr in Edinburgh, Schottland von 14. — 17. September 2018 statt. Ausrichter Paul Bennett hat Chuck Moore eingeladen.

Die Tagung 2019 wird in Worms stattfinden und wird von Ewald Rieger organisiert. Termin und Ort werden rechtzeitig nach Abschluss der Planungen bekannt gegeben.

Ende der Versammlung um 11:26 Uhr.

Essen, den 5.6.2018

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**$_p$

Termine

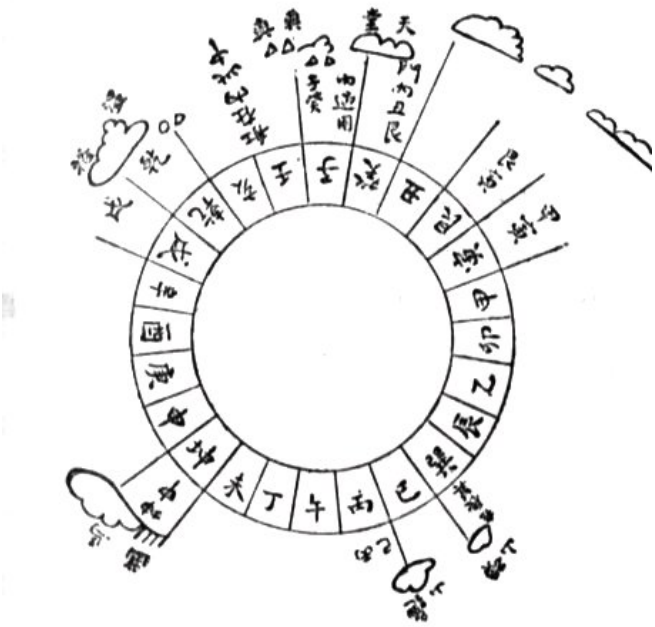
Donnerstags ab 20:00 Uhr
Forth–Chat net2o `forth@bernd` mit dem Key `keysearch kQusJ`, voller Key:
`kQusJzA;7*?t=uy@X}1GWr!+0qqp_Cn176t4(dQ*`

14.–16.09.2018 Maker Faire Hannover
`http://www.makerfairehannover.com`

14.–17.09.2018 EuroForth in Edinburgh
`http://www.euroforth.org`

27.–30.12.2018 35C3 Leipzig
`http://events.ccc.de/congress`

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.

EuroForth in Edinburgh, September 14.–17. 2018

Forth Conference (including the optional Forth Day) is Friday 14th September to Monday 17th September. Banquet night will be Saturday 15th September.

The Standards Meeting will begin on Wednesday 12th September and finish on Friday 14th September. Usual midday to midday timetable.

The venue will be the DoubleTree Hilton Queensferry Hotel on the North Shore of the Firth of Forth. There is a local railway station nearby and Edinburgh Airport is reachable through many other hubs within Europe.

I have invited Chuck and he initially said he would attend. If anyone wants to sponsor Chuck's visit to pay his airfare and stay with us, then I would be interested in them contacting me to make it known. I think it would be nice if we can do a suitable gift for him too.

Paul Bennett

Preise

	Delegate	Partner
Standards Meeting (2 nights)	£390.00	£100.00
Conference Only (3 nights)	£600.00	£150.00
Standards and Conference	£975.00	£250.00

Conference only

Friday afternoon	£46.00
Saturday all day	£53.00
Sunday morning	£46.00
Friday evening meal	£20.00
Saturday Night Conference Banquet	£35.00
Conference only total	£200.00

Link

`http://www.euroforth.org`

Abbildung 1: Das Stadtwappen von Edinburgh

Abbildung 2: Skyline von Edinburg Andrew Colin

Abbildung 3: Forth Bridge Reinhard Berlin

Abbildung 5: Forth Road Bridge

Abbildung 4: DoubleTree Hilton Queensferry Hotel