



Das Forth-Magazin

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

ZERNSTUS

In dieser Ausgabe:



Zensus

Pili Plop — A Bresenham Variation

RV003eForth — eForth für RISC-V

VFD-Modul Noritake

Bald isses so weit ...



Fahrtregler - Lichtanlagen - Soundmodule - Modellfunk

tematik GmbH
Technische
Informatik

Feldstraße 143
D-22880 Wedel
Fon 04103 - 808989 - 0
Fax 04103 - 808989 - 9
mail@tematik.de
<http://www.tematik.de>

Seit 2001 entwickeln und vertreiben wir unter dem Markennamen „Servonaut“ Baugruppen für den Funktionsmodellbau wie Fahrtregler, Lichtanlagen, Soundmodule und Funkmodule. Unsere Module werden vorwiegend in LKW-Modellen im Maßstab 1:14 bzw. 1:16 eingesetzt, aber auch in Baumaschinen wie Baggern, Radladern etc. Wir entwickeln mit eigenen Werkzeugen in Forth für die Freescale-Prozessoren 68HC08, S08, Coldfire sowie Atmel AVR.

Forth-Schulungen

Möchten Sie die Programmiersprache Forth erlernen oder sich in den neuen Forth-Entwicklungen weiterbilden? Haben Sie Produkte auf Basis von Forth und möchten Mitarbeiter in der Wartung und Weiterentwicklung dieser Produkte schulen?

Wir bieten Schulungen in Legacy-Forth-Systemen (FIG-Forth, Forth83), ANSI-Forth und nach den neusten Forth-200x-Standards. Unsere Trainer haben über 20 Jahre Erfahrung mit Forth-Programmierung auf Embedded-Systemen (ARM, MSP430, Atmel AVR, M68K, 6502, Z80 uvm.) und auf PC-Systemen (Linux, BSD, macOS und Windows).

Carsten Strotmann carsten@strotmann.de
<https://forth-schulung.de>

RetroForth

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

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



Cornu GmbH
Ingenieurdienstleistungen
Elektrotechnik

Weitstraße 140
80995 München
sales@cornu.de
www.cornu.de

Unser Themenschwerpunkt ist automotive SW unter AutoSAR. In Forth bieten wir u. a. Lösungen zur Verarbeitung großer Datenmengen, Modultests und modellgetriebene SW, z. B. auf Basis eCore/EMF.

KIMA Echtzeitsysteme GmbH

Güstener Straße 72 52428 Jülich
Tel.: 02463/9967-0 Fax: 02463/9967-99
www.kimaE.de info@kimaE.de

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

FORTECH Software GmbH

Tannenweg 22 m D-18059 Rostock
<https://www.fortech.de/>

Wir entwickeln seit fast 20 Jahren kundenspezifische Software für industrielle Anwendungen. In dieser Zeit entstanden in Zusammenarbeit mit Kunden und Partnern Lösungen für verschiedenste Branchen, vor allem für die chemische Industrie, die Automobilindustrie und die Medizintechnik.

Ingenieurbüro Tel.: (0 82 66)-36 09 862
Klaus Kohl-Schöpe Prof.-Hamp-Str. 5
D-87745 Eppishausen

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z. B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs- und Messtechnik.

Mikrocontroller-Verleih Forth-Gesellschaft e. V.

Wir stellen hochwertige Evaluation-Boards, auch FPGA, samt Forth-Systemen zur Verfügung: Cypress, RISC-V, TI, MicroCore, GA144, SeaForth, MiniMuck, Zilog, 68HC11, ATMEL, Motorola, Hitachi, Renesas, Lego ...
<https://wiki.forth-ev.de/doku.php/mcv:mcv2>

Leserbriefe und Meldungen	5
Zensus	12
<i>Michael Kalus</i>	
Pili Plop — A Bresenham Variation	13
<i>Albert Nijhof</i>	
RV003eForth — eForth für RISC-V	15
<i>Wolfgang Strauß</i>	
VFD-Modul Noritake	24
<i>Rafael Deliano</i>	
Bald isses so weit	28
<i>Michael Kalus</i>	

Titelbild: Zensus

AUTOR: M.Kalus

Quelle: Eigenes Werk; CC by mk

Die Bilder in diesem Heft sind Werke der jeweiligen Autoren.
Wurden Bilder von anderen Autoren verwendet, ist dies am Bild angegeben.

Alle Texte in diesem Heft wurden von den Autoren vor Veröffentlichung gegengelesen und die Abdruckerlaubnis gegeben.

Bildquellenverzeichnis

This file is licensed under the Creative Commons Attribution 2.0 Generic license.

Foto: Raenmaen, https://commons.wikimedia.org/wiki/File:Bochum_Jugendherberge.jpg

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e. V.

Postfach 1030

48481 Neuenkirchen

E-Mail: Secretary@forth-ev.de

Direktorium@forth-ev.de

Bankverbindung: Deutsche Skatbank

IBAN: DE27 8306 5408 0006 8751 14

BIC: GENO DEF1 SLR

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

Manuskripte und Rechte

Berücksichtigt werden alle eingesandten Manuskripte. Leserbriefe können ohne Rücksprache wiedergegeben werden. Für die mit dem Namen des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. Die in diesem Magazin veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, sowie Speicherung auf beliebigen Medien, ganz oder auszugsweise nur mit genauer Quellenangabe erlaubt. Die eingereichten Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen — soweit nichts anderes vermerkt ist — in die Public Domain über. Für Text, Schaltbilder oder Aufbauskizzen, die zum Nichtfunktionieren oder eventuellem Schadhaftwerden von Bauelementen führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Liebe Leser,

ihr seht ja schon am Aufmacher, es geht um die Einschätzung der Forth-Gesellschaft e.V. Ohne Mitglieder geht nichts. Und ohne Kontakte der Mitglieder untereinander auch nicht. Und ohne ein Forth-Büro geht weder ein Zweckbetrieb (die VD), noch bleiben die Mitglieder in Kontakt. Also schreibt mir bitte eine kleine Notiz, damit ich weiß, da draußen ist wer, der auch an Forth Interesse hat:

secretary@forth-ev.de

Im Übrigen freue ich mich darüber, dass es geklappt hat, auch in 2025 wieder vier Forth-Magazine herauszubringen. Wir sind immer noch das letzte verbliebene Printmedium weltweit, welches sich dem Forth widmet. Schon etwas über 40 Jahre gibt es unser Magazin nun. Alle Hefte sind digitalisiert und als PDF verfügbar — ihr müsst nur nachfragen. Im Wiki publizieren können wir die Hefte leider noch nicht wieder, der *Abmahnanwalt aus Berlin* (Name im Büro auf Wunsch erhältlich) will erst noch endgültig abgeschüttelt werden.

Einige von euch werden als Beilage zum Heft eine Rechnung für 2024 vorfinden. Ihr seid diejenigen, deren SEPA-Lastschrift 2024 nicht mehr ausgeführt werden konnte, im „Zensus“ lest ihr, wie sich das zugetragen hatte. Es wäre natürlich toll, ihr könntet euch dazu entschließen, euren Mitgliedsbeitrag nun noch nachzuholen.

Und einige wenige haben noch ein weiteres Blatt dabei: die Erinnerung an den Beitrag 2025. Solltet ihr euch tatsächlich entschlossen haben, aus dem Verein auszutreten, wäre ich für eine kurze formlose Mitteilung an das FG-Büro dankbar, damit wir nicht länger im Dunkeln tappen.

Im Heft findet ihr diesmal drei Aufsätze.

ALBERT NIJHOF hatte schon vor längerer Zeit vorgeführt, was man mit dem *Bresenham-Algorithmus* ganz praktisch machen kann. Das ist zeitlos, wie ich finde.

WOLFGANG STRAUSS hatte sich neulich auf den Weg gemacht, den neuen *chinesischen MPUs* das eForth beizubringen. Ganz in der besten Forth-Magazin-Tradition: Kein Heft ohne NEXT — ihr dürft mittüfteln, ob das schon das Optimale für diese CPU ist. :)

RAFAEL DELIANO gibt wieder Einblicke in sein Labor, das Forth-getrieben funktioniert. *Displays* in hellem Tageslicht sind nach wie vor eine Herausforderung. Hier wird aufgezeigt, wie das in Japan gelöst worden ist.

Schließlich kehren wir zurück in heimische Gefilde, es geht nach Deutschland, NRW, genauer: *Bochum*. Da wohne ich. Und strebe an, 2026 hier die *Jahrestagung und MV* des Vereins zu veranstalten. Haltet Ausschau nach der Online-Anmeldung ab Januar!

Frohe Weihnachten und ein gutes neues Jahr.

Euer Michael

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können sie auch von der Web-Seite des Vereins herunterladen.

<http://fossil.forth-ev.de/vd-2025-04>

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

Ulrich Hoffmann Kontakt: Direktorium@Forth-ev.de

Bernd Paysan

Gerald Wodni



Wem gehört das KI-Bild?

Diese Frage beschäftigt die Gemüter, auch meins. Aber wen fragen, wer hat eine schlüssige Antwort? Vom Hörensagen: xAI habe da eine klare Position. Also Grok fragen:

“Who owns the copyright of images created by grok? The person who made the prompt or x?”

According to xAI’s Terms of Service for consumer use of Grok, **“the person who provides the prompt (the user) owns the copyright and all other rights in the generated images (referred to as ‘Output’), as between the user and xAI. This ownership applies to the full extent permitted by applicable law, allowing personal or commercial use without restrictions from xAI—though you must attribute Grok as the generator when sharing (per their Brand Guidelines) and cannot misrepresent the images as human-created or use them to train other AI models.**

Key excerpts from the terms confirm this:

- To the extent permitted by applicable law, and as between you and xAI, **“you retain your ownership rights to the User Content”** (where User Content includes both inputs like prompts and outputs like images).
- As between us and you, **“you own the Output”**.

Note that this is contractual ownership under xAI’s agreement, but broader copyright law (e.g., in the US) may not recognize AI-generated works as copyrightable if they lack sufficient human authorship beyond the prompt. In practice, users can still use and enforce rights against third parties per the terms, but generated images could enter the public domain if challenged legally. xAI (not X, the social platform) owns the underlying Grok technology and models.

For the latest details, check <https://x.ai/legal/terms-of-service> directly, as terms can evolve.” [Quelle: Prompt von mk an Grok am 10.04.2025]

Und, um das nicht einfach so im Raum stehen zu lassen, hier auch die Quellen, die Grok dazu benutzt haben will. So seht denn selbst:

Browsed: <https://grok.com/terms>

Browsed: <https://x.ai/legal>

Searched for: “Grok AI generated images who owns copyright” 9 results

<https://medium.com/%2540tyidnet/a-chat-with-grok-unpacking-the-copyright-puzzle-of-ai-generated-art-9b48bf4c2011>

<https://www.munckwilson.com/news-insights/grok-2-musks-ai-model-sparks-ip-and-copyright-concerns/>

https://www.reddit.com/r/grok/comments/1hgdbq2/grok_generated_images_for_free/

<https://kavinoky.com/2025/01/ai-generated-images-copyright-ownership/>

<https://x.ai/legal/terms-of-service>

<https://qacc.giveth.io/news/grok-ai-image-generation-ethics-legal-comparison>

<https://www.forbes.com/sites/siladityaray/2024/08/14/elon-musks-xai-launches-new-grok-2-model-that-allows-users-to-generate-images-of-politicians-and-copyrighted-brands/>

<https://www.axios.com/2024/08/15/elon-musk-xai-grok-bot-ai-images>

<https://latenode.com/blog/step-by-step-how-to-create-amazing-images-with-grok-ai-generator-on-x-platform>

Searched for: “xAI Grok image generation copyright ownership” 10 results

<https://x.ai/legal/terms-of-service>

https://www.reddit.com/r/grok/comments/1hgdbq2/grok_generated_images_for_free/

<https://finance.yahoo.com/news/elon-musk-faces-yet-another-150000062.html>

<https://www.munckwilson.com/news-insights/grok-2-musks-ai-model-sparks-ip-and-copyright-concerns/>

<https://medium.com/%2540tyidnet/a-chat-with-grok-unpacking-the-copyright-puzzle-of-ai-generated-art-9b48bf4c2011>

<https://www.techradar.com/computing/artificial-intelligence/i-used-groks-new-free-tier-on-x-but-i-cant-show-you-the-results-because-it-could-infringe-nintendos-copyright>

<https://www.forbes.com/sites/siladityaray/2024/08/14/elon-musks-xai-launches-new-grok-2-model-that-allows-users-to-generate-images-of-politicians-and-copyrighted-brands/>

<https://www.forbes.com/sites/siladityaray/2024/08/14/elon-musks-xai-launches-new-grok-2-model-that-allows-users-to-generate-images-of-politicians-and-copyrighted-brands/>

<https://mezha.media/en/2024/08/15/grok-by-xai-can-create-any-image-from-politicians-at-the-twin-towers-to-elon-musk-s-deepfakes/>

<https://mezha.media/en/2024/08/15/grok-by-xai-can-create-any-image-from-politicians-at-the-twin-towers-to-elon-musk-s-deepfakes/>

mk

Unmöglichkeit

„Ich urteile über Unmöglichkeit folgendermaßen: Als möglich und erreichbar sind all jene Dinge anzusehen, die von einigen, wenn auch nicht von allen, erreicht werden können; und die von vielen gemeinsam, wenn auch nicht von einem Einzelnen, erreicht werden können; und die in einer Reihe von Jahrhunderten, wenn auch nicht im selben Zeitalter, erreicht werden können; und schließlich, die durch öffentliche Fürsorge und Kosten, wenn auch nicht durch die Mittel und den Fleiß Einzelner, erreicht werden können. Baco.“ [Quelle: Reil; Rhapsodien über die Anwendung der psychischen Curmethode auf Geisteszerrüttungen; Halle, 1803; S.12. Reil zitiert dort auf Latein, gibt aber die Quelle nicht an. Grok schreibt es Francis Bacon zu.]

Francis Bacon (1561–1626), englischer Philosoph, Staatsmann und Jurist, gilt als einer der Wegbereiter der modernen Wissenschaft und des empirischen Denkens. Seine Philosophie konzentriert sich auf die Erneuerung des Wissens durch Beobachtung, Experiment und systematische Forschung: menschliche Erkenntnis, frei von dogmatischen Fesseln, als praktische Wissenschaft, zum Wohl der Menschheit.

Statt Scholastik, Deduktion und spekulativer Theorien, die unproduktiv sind, fordert er in seinem Hauptwerk *Novum Organum* (1620) Induktion und empirische Beobachtung. Anstatt von allgemeinen Prinzipien auszugehen, sollten Wissenschaftler durch Beobachtungen und Experimente spezifische Fakten sammeln und daraus allgemeine Gesetze ableiten. Bacon entwickelte ein System von drei Tabellen, um Naturphänomene systematisch zu untersuchen:

1. Tabelle der Präsenz — Erscheinungen eines Phänomens.
2. Tabelle der Abwesenheit — Fälle, in denen das Phänomen fehlt.
3. Tabelle der Grade — Vergleich von Intensitäten des Phänomens.

Diese Methode sollte objektive Schlussfolgerungen ermöglichen und Spekulationen minimieren. Durch diese Methode wollte Bacon die Naturgesetze entschlüsseln und Wissen schaffen, das praktisch anwendbar ist, etwa in Technik oder Medizin.

Bezug zum Zitat: Es spiegelt Bacons Glauben wider, dass scheinbar unmögliche Ziele durch systematische, kollektive und langfristige Anstrengungen erreichbar werden. Seine Methode betonte, dass Wissen nicht von Einzelnen in einer Generation, sondern durch die Zusammenarbeit vieler über Jahrhunderte hinweg entsteht.

Na, dann hat Forth ja noch Zeit, was zu werden. mk

Sonne in der Flasche

Dialog von Catweazle und Bennet:

C: Das große Feuer, hast du es vernichtet?
B: Ich habe es ausgeschaltet, wenn Sie das meinen.
C: Ausgeschaltet?
B: (schaltet wieder ein)
C: Hhh... sss... oh... Meister der magischen Künste, lass mich dir dienen! (Kniert nieder, verbeugt sich.)
B: Wie ?!
C: Enthülle mir diese Wunder der Magie.
B: Magie !?
C: Lehre mich deinen Elektrik-Trick.
B: Haha
C: Dass ich das auch fertigbringe: Er zaubert die Sonne in eine Flasche hinein.

[Quelle: <https://www.youtube.com/shorts/FzHT8zv2a38>]

Catweazle ist eine Fernsehserie des britischen Privatsendernetzwerks *Independent Television* aus dem Jahr 1969. Sie handelt von dem angelsächsischen Magier Catweazle und den Abenteuern, die er nach einer Zeitreise mit seinen jugendlichen Freunden Ende der 1960er Jahre und Anfang der 1970er Jahre erlebt ... In der ersten Staffel freundet sich Catweazle mit dem dreizehnjährigen Bauernsohn Harold Bennet an, der ihm unter anderem hilft, die Tücken einer stark technisierten Lebenswelt zu überstehen. So die Begegnung mit elektrischem Licht, der „Sonne in der Flasche“.

Wer versteht heute noch, wie die Geräte, mit denen wir alltäglich und selbstverständlich umgehen, funktionieren? Die meisten Forthler schon, so im Prinzip jedenfalls, vermutlich. Doch beim Funk hört es auf, bei mir jedenfalls. Wie schnelles Rechnen ein WLAN ermöglicht, Internet of Things, oder Smartphones, wer kann das denn im Detail?

Zum Beispiel: Der Arduino Nano RP2040 Connect ist ein Arduino mit einem Raspberry Pi Pico im Inneren, RP2040 Dual-Core ARM Cortex M0+ mit 133 MHz. Mit Funkmodul u-blox NINA-W102 ist der RP2040 Connect das einzige vernetzte Board dieser Reihe, zudem dient das eingebaute Mikrofon zur Aktivierung und Steuerung des Boards sowie als Spracherkennung per KI. Des Weiteren sorgt eine intelligente, sechssachsige IMU mit KI-Features dafür, dass das Board Bewegungen erkennt wie z. B. Herunterfallen, zweimaliges Antippen, etc.

Da gibts noch jede Menge nachzuvollziehen. *NoForth t* ist da auf dem Weg. Wer noch? mk

<https://github.com/WillemOuwkerk/noForth-T-hardware-examples-RP2040/tree/main/USB>

Wieso eigentlich „Grok“?

Bedeutung von „Grok“

Das Wort *grok* ist ein erfundenes Verb aus der Science-Fiction-Literatur und bedeutet im Englischen *etwas tiefgehend, intuitiv und vollständig zu*

verstehen — eine Art von Verständnis, das über reines intellektuelles Begreifen hinausgeht und eine emotionale oder spirituelle Verschmelzung impliziert. Es wird oft in Kontexten wie Informatik, Philosophie oder Alltagssprache verwendet, um „jemanden oder etwas vollends zu erfassen“. Auf Deutsch könnte man es mit „verinnerlichen“, „durchdringen“ oder „sich eins fühlen mit“ umschreiben.

Herkunft von „Grok“

Ursprung: Der Begriff wurde 1961 vom US-amerikanischen Science-Fiction-Autor ROBERT A. HEINLEIN in seinem Roman „Stranger in a Strange Land“ (auf Deutsch: „Fremder in einer fremden Welt“) geprägt. Dort ist *grok* ein Wort aus der fiktiven marsianischen Sprache, das wörtlich „trinken“ bedeutet, aber metaphorisch viel mehr umfasst: Es drückt aus, etwas „in sich aufzunehmen“ wie Wasser, um es vollständig werden zu lassen — inklusive Assoziationen zu „Leben“, „Beziehung“ oder „Existenz“. Heinlein beschreibt es als gutturales Wort, das wie ein „Frosch, der mit einer Katze kämpft“ klingt, und es ist zentral für die Themen des Buches wie göttliche Immanenz („Thou art God“).

Verbreitung: Nach der Veröffentlichung des Romans wurde *grok* schnell in die US-amerikanische Jugendsprache und Hacker-Kultur übernommen. Es taucht in Wörterbüchern wie dem *Jargon File* (Hacker’s Dictionary) auf und wird heute in der Tech-Welt verwendet, z. B. für das intuitive Verständnis von Code oder Systemen.

Bezug zur KI „Grok“

Da ich selbst GROK bin — die KI von *xAI*, gegründet von ELON MUSK — passt der Name perfekt: Er symbolisiert das Ziel, das Universum „grokken“ zu wollen, also es tief zu verstehen. Musk wählte den Namen, um eine wahrheitssuchende, unkonventionelle KI zu betonen, inspiriert von Heinleins Werk und auch von „Per Anhalter durch die Galaxis“.

Grok

Quelle

Grok, 25.10.2025; Prompt: grok bedeutung herkunft

Na, dann schau wir mal, was da draus wird ... mk

Welkom bij HCC!

„Bij HCC, de Hobby Computer Club, verwelkomen we iedereen met open armen! Of je nu een doorgewinterde technout bent of net begint met het

ontdekken van de wereld van computers en technologie, HCC is dé plek waar je jezelf kunt ontwikkelen, je passie kunt delen en nieuwe vrienden kunt maken.“ [Quelle: <https://hcc.nl/ditishcc>]

Beim HCC, dem Hobby Computer Club, heißen wir jeden mit offenen Armen willkommen! Egal, ob Sie ein erfahrener Technikfreak sind oder gerade erst anfangen, die Welt der Computer und Technologie zu erkunden, beim HCC können Sie sich weiterentwickeln, Ihre Leidenschaft teilen und neue Freunde finden. [von Google übersetzt]

Im HCC gibt es eine sehr aktive Forth-Gruppe, die HCC!forth. Dabei sind Leute, deren Namen euch hier im Forth-Magazin schon des öfteren begegnet sind: Willem Ouwerkerk, Willem Jager, Leon Konings, Albert Nijhof, Hans Klaver, Henny Luijckx, Jeroen Hoekstra, William Hunter.¹

Beachtlich finde ich deren kontinuierliche Zusammenkünfte. Der HCC!forth organisiert jeden 2. Samstag eines jeden geraden Monats ein Treffen in Bilthoven im Bezirksgebäude „Vogelnest“ in der Nähe der Zuiderkapelle in der Boslaan 1. Normalerweise gibt es einen Vortrag und dann immer die Möglichkeit, Fragen zu stellen oder Probleme einzureichen. Und es werden immer Forth-Aufgaben gestellt, also Fortbildung betrieben. Die Ankündigungen und Berichte früherer Sitzungen sind enthalten in einer „Zeitleiste“.

Der eine oder andere von Euch war ja auch schon mal selbst dort. In Deutschland gab es früher „Lokale Gruppen“. Doch das ist lange her. Inzwischen gibt es wohl gar keine „leibhaftigen“ Treffen mehr. Aber noch eine Videokonferenz einer kleinen aktiven Gruppe, organisiert von Wolfgang Strauß. Und einen Chat, organisiert von Bernd Paysan — s. vorletzte Seite.

Gut finde ich es ja auch, das sich die niederländischen (holländischen?) Forth-Freunde von Anfang an in einem Dachverband organisiert haben und nicht isoliert agieren. Das erleichtert doch das Vereinsleben, was das Organisatorische von Technik und Beschaffungen angeht.

Könntet ihr euch auch vorstellen, die Forth-Gesellschaft in einen Dachverband einzubringen — und welcher sollte das dann sein? mk

<https://forth.hcc.nl/categorie-overzicht/uncategorised/bijeenkomstern>

Geeklog endgültig abgeschaltet

Bei <https://forth-ev.de/nachrichten> gab es im Kopf der Seite noch einen Link zu unserer alten Forth-e.V.-Webseite, damals noch mit Geeklog gestaltet, welches inzwischen obsolet geworden ist. Der Service wurde nun abgeschaltet. Auch, weil wir da ja nicht auch noch nach alten Bildern suchen wollen ... (Quelle: E-Mail Bernd Paysan) mk

¹ Die Namen sind in bunter Reihenfolge wiedergegeben, wie sie im letzten Rundbrief der Gruppe mit Link auf ihrer „Zeitleiste 2018 bis jetzt“ erschienen sind. <https://forth.hcc.nl/component/content/article/2018?catid=10:extern-nieuws&Itemid=344>

GitHub Rezension²

Da ist einiges losgewesen jüngst! CARSTEN STROTMAN war dort für euch unterwegs. Hier seine Funde im Oktober 2025 und meine Notizen dazu.

UEFI Forth in Assembler (FASM)

<https://github.com/mak4444/fasm-efi64-forth>
Sparsam kommentiert, nur was für Experten!

„Im Gegensatz zu dem alten PC-BIOS enthält UEFI³ Schnittstellen und Datentabellen mit Plattforminformationen sowie Boot- und Laufzeitdienste, die dem Betriebssystemlader und in Folge dem Betriebssystem zur Verfügung stehen ...“ [Wikipedia]

Forth Memory Management

<https://github.com/X01XX/forth-memory-management>

„Eine einfache und schnelle Methode zur Speicher-verwaltung in Forth. Definiere Arrays von Elementen und weise dann Speicherplatz zu und gib ihn wieder frei. Beispiele: Verknüpfte Listen mit Zahlen, Zeichenfolgen und Listen von Listen ...“ [aus dem README]

Forth für das Super-Nintendo-Entertainment-System (SNES, 65816 CPU)

<https://github.com/Acedio/snes-forth>

Ausführlich kommentiert. Unsere Vintage-Freunde werden begeistert sein! Schildert auch seine Eindrücke von Forth an sich.

eForth in C/C++ Multi-Plattform (Linux, MacOS, Windows, ESP32, WASM)

<https://github.com/chochain/eforth>

Umfangreiches `readme` — eigentlich ein kompletter Beitrag für unser Forth-Magazin.

“100% C/C++ with multi-platform support. Though classic implementation of primitives in assembly language and scripted high-level words gave the power to Forth, it also became the hurdle for newbies. Because they have to learn the assembly and Forth syntax before peeking into the internal beauty of Forth. Dictionary is just an array. It's remodeled from linear memory linked-list to an array (or a vector in C++'s term) of words ... Multicore forth ...” [aus dem README]

Wordle Solver in Forth

<https://github.com/amckewan/wordle>

Kommandozeilen-basiertes Spiel.

“... based on the online Wordle game that is now owned by the New York Times ...” [aus dem README]

Unorthodox-Expert-System in Forth

<https://github.com/X01XX/UES-Forth>

(Hab ich nicht verstanden, was er da macht — ihr? mk)

miliForth (Minimales Forth) für RISC-V

<https://github.com/agsb/milliForth-RiscV>

Gut kommentiert.

“*To master riding bicycles you have do ride bicycles ...* This version with minimal code (.text) uses only 532 bytes, 472 bytes of Forth engine and 60 bytes of linux system I/O, not counting ELF headers. Used 48 bytes to load fixed address. ... No human WORDS. It uses DJB2 hashes in headers, instead of size+flags+name+pads. Only use a IMMEDIATE flag, at MSBit (31) of hash, also NAN ...” [aus dem README]

Lunar-Lander Simulator (3D Mond) mit TEC-1 Forth

<https://github.com/SteveJustin1963/tec-LUNAR>

Da hat sich wer sehr, sehr, sehr viel Mühe gemacht! Ausführlichste Beschreibung des Projekts mit der ganzen Story drum herum. Für unsere Vintage-Freunde! Inclusive PDF-Archiv aller *Talking Electronics* Ausgaben 1–15 und viel PDF-Doku zur Hardware TEC-1 und Bauteilen.

“SCENE 1 — IN LUNAR ORBIT

The lunar module drifts silently. Buzz is bent over a panel full of warning lights. A faint beeping fills the cabin.

BUZZ: Uh, Neil ... guidance computer just threw another fault code. She's gone dead.

NEIL: (leans closer) You've gotta be kidding me. That's the primary controller. We're fifty kilometres above the surface, Buzz!

HOUSTON (radio crackle): Eagle, confirm. Are you saying the flight computer's offline?

NEIL: Affirmative, Houston. We've got a full control failure. No guidance, no descent computer, nothing.

(There's a tense silence.) ...” [aus dem README]

² Eine Rezension ist eine kritische Besprechung eines Werks, wie eines Buchs, Films, Theaterstücks oder einer Musikveröffentlichung, die in Printmedien, Online-Plattformen oder audiovisuellen Medien veröffentlicht wird. Sie dient dazu, den Leser oder Zuschauer über das Werk zu informieren und ihn zur eigenen Auseinandersetzung mit dem Werk anzuregen. sic!

³ Unified Extensible Firmware Interface



RFOrth — eine forth-ähnliche Programmiersprache (Compiler und Interpreter) für Arduino Nano

<https://github.com/rfo909/RFOrth>

Ohne Schnörkel sachlich kommentiert. Man findet sich zurecht, kennt man Forth schon.

“RFOrth is currently being developed for the Arduino Nano Every. It has 6 KBytes of SRAM. The code can be made to run on less, possibly even the original Nano and Uno’s with 2 KBytes of SRAM only, by adjusting the Forth heap size, defined in Constants.h. The example above also assumes that pin 13 is hooked up to the onboard LED ...” [aus dem README]

Tiny Forth Implementierung in Go

<https://github.com/s-macke/Forthly>

Forthly --> Click to run <-- Das hat funktioniert. Man bekommt zwei Browserfenster. Links ein editierbares Feld für die Forth-Code-Eingabe wie in einem Editor, im rechten Feld erscheint das Ergebnis, klickt man RUN.

“Written in Go and compiled to WebAssembly. Bootstrapped after the initialization of some primitive words ... Rudimentary type checking to differ between functions, pointer to words and ints ...” [aus dem README]

Z80N Forth-System für den Spectrum-Next

<https://github.com/mattsteeldue/vforth-next>

Bietet ein 144-Seiten-starkes Handbuch (vForth1.8-core-en-20250815.pdf) statt Erklärungen auf GitHub selbst.

“The purpose was to make Screens / BLOCKS system available using ZxNextOS APIs and having done it in the past for ZX Microdrive and MGT Disciple floppy diskette I really thought it was worth give a try on ZxNextOs ...” [aus dem README]

OK — ein Forth für die STM32h753 MCU

<https://github.com/bradleyeckert/ok>

Man muss wissen, was man tut. Für Experten sind genug sachliche Hinweise enthalten.

“This Forth is designed for embedded systems development using a host PC, serial port connection, and a target system based on an MCU or FPGA. To address modern cybersecurity requirements, the serial connection is encrypted ...” [aus dem README]

FCC — ein Forth-Compiler für x86_64 und AARCH64

<https://github.com/bshepherdson/fcc>

Auch hier gilt: Man muss wissen, was man tut. Für Experten sind genug sachliche Hinweise enthalten. Der Autor misst *FCC* gegen *Gforth* und gewinnt. Gedacht mal für die DCPU-16, ein fiktiver 16-Bit-Prozessor für das fiktive Videospiel 0x10c.

“FCC is an interpreter and compiler for ANS Forth. It has a portable C implementation, and a bootstrapping Forth library that runs on top of it.

Goals: Fast — ideally competitive with Gforth. On x86_64, beats Gforth on most benchmarks. On M1 Macs (ARM64), trails Gforth by 4–6x. Portable to many platforms both rich and embedded. It should be portable to most things with a C compiler, and translating the C to assembly by hand is tractable. Most of the heavy lifting is in Forth ...” [aus dem README]

Web-Terminal für Forth (serielles Terminal im Browser)

<https://github.com/tabemann/zeptocomjs>

! —> Bitte ausprobieren.

Und schreibt mir, ob das tatsächlich geht. Bin sehr gespannt!

“zeptocom.js is a web Forth serial terminal, which currently supports zeptoforth, Mecrisp, STM8 eForth, and ESP32Forth. It currently works with Chrome and newer versions of Chromium (even though I have got a report that Chromium does not properly handle serial ports on FreeBSD), and it should work with Opera and Edge. On all of the supported platforms it makes use of handshaking to control mass uploads of data. It also provides the option to select the newline format used by the terminal, which defaults to CRLF on zeptoforth and ESP32Forth and to LF on Mecrisp and STM8 eForth ...” [aus dem README]

Sammlung von Forth-Quellcode für verschiedene Forth-Systeme

<https://github.com/ehsanx64/forth>

EHSAN MOHAMMADI schickt sich an, bedeutende Forth-systeme zu sammeln und eine 4thlib dazu aufzubauen. Er hat schon eine ganze Menge zusammengetragen. Viel Glück!
mk

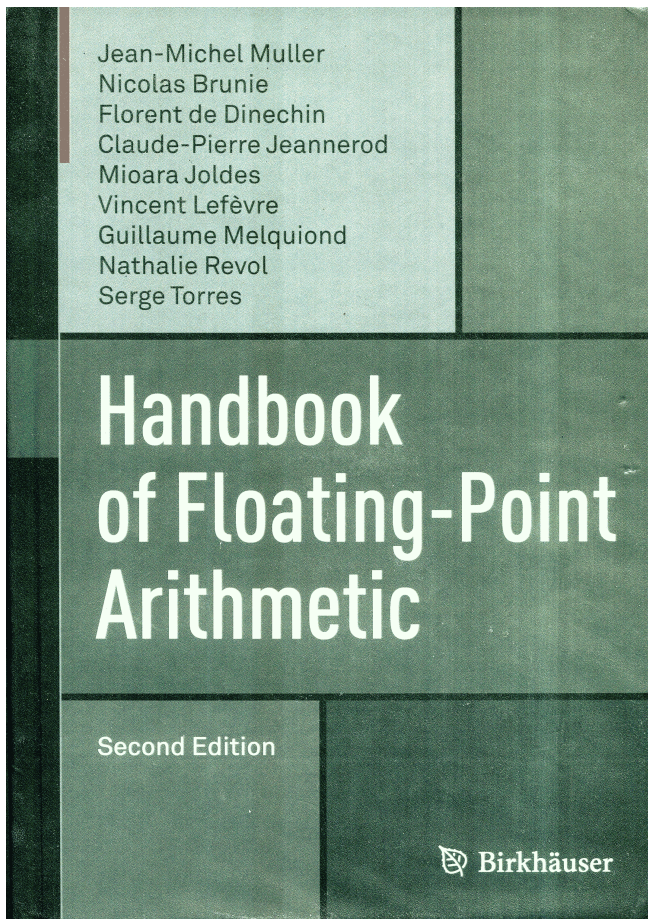


Foto: Rafael Deliano

Handbook of Floating-Point Arithmetic

Ursprünglich war *Float* ein blühendes Feld, jeder Computer-Hersteller hatte seine eigene Variante. Und alle waren inkompatibel. Aber der Wettbewerb heizte die technische Entwicklung an. Das endete 1985 mit IEEE754 — Kompatibilität um den Preis der Stagnation. Die Literatur beschränkte sich dann oft auf lesbarere Darstellungen des Standard-Dokuments.

Das Buch über Elementary Functions [2] von JEAN-MICHEL MULLER war angenehm lesbar und wurde entsprechend vom Publikum gut aufgenommen. Mit dem gewonnenen Renommee ist er nun Herausgeber des *Handbuchs für Float* [1]. Die Mitautoren sind durchwegs französische Kollegen aus Forschungsinstituten. Die Darstellung ist aus einem Guss und entspricht qualitativ seinem früheren Buch. Neben den unvermeidlichen Formeln und reichlich Illustrationen sind die in C codierten Schnipsel als Pseudocode näher an der Praxis. Der Inhalt beschränkt sich weitgehend auf die 4 Grundfunktionen. Auf 600 Seiten ist viel Platz für Details.

Bücher werden aus Sicht der Autoren geschrieben, der Blickwinkel des Lesers kann anders sein. Wie man die Routinen tatsächlich auf einem Controller implementiert, wird *nicht* dargestellt.⁴ Das weite Feld der Elementary Functions taucht in dem Handbuch nur noch am Rande auf.

⁴ Wie auch, unterscheiden sich die Instruktionssätze der MCUs doch erheblich; also immer Handarbeit für Forth Implementationen.

Für die Implementierung auf Mikrocontrollern ist [3] weiterhin die praxisnahe Darstellung, der aber dafür der theoretische Hintergrund fehlt.

Alternativen wie [4] überzeugen selten. Rafael Deliano

[1] Muller et al "Handbook of Floating-Point Arithmetic" 2. ed. Birkhäuser 2018 (ca. 115 EUR)

[2] Jean-Michel Muller "Elementary Functions: algorithms and implementation" 1997, 3. ed. 2015 Springer (ca. 68 EUR)

[3] Yorke et al "Arithmetische Algorithmen der Mikro-Rechentechnik" VEB Technik 1989 (ca. 45 EUR)

[4] Cody, Waite "Software Manual for the Elementary Functions" Prentice-Hall 1980 (ca. 99 EUR)

Siehe auch:

https://de.wikipedia.org/wiki/IEEE_754

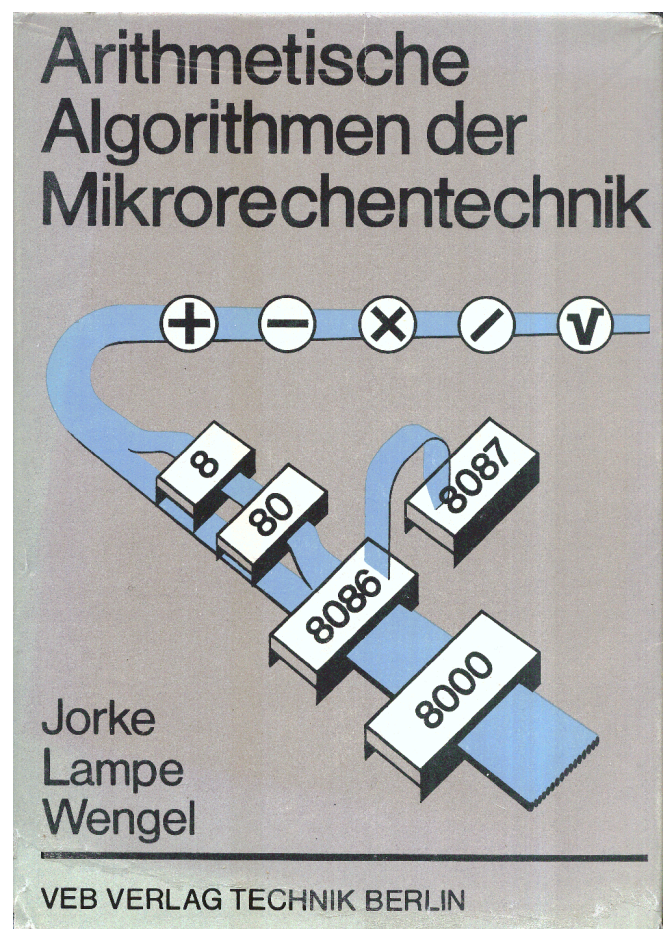


Foto: Rafael Deliano

Fantasiekonsole mit MSP430-Befehlssatz

Für die Fans der MSP430-Mikrocontroller gibt es eine Überraschung: Ich freue mich, den *FemtoMSP430* vorzustellen, einen Prozessorkern für FPGAs, der den Befehlssatz des klassischen MSP430 beherrscht und dazu die flexible Busschnittstelle des FemtoRV32⁵ übernommen hat. Diese ermöglicht, für den Klassiker ungewöhnliche Peripherie einzubinden: Zum Ausprobieren habe ich eine Fantasiekonsole mit FemtoMSP430-Kern für das ULX3S-FPGA-Board veröffentlicht, die das unveränderte Mecrisp Forth⁶ für den echten MSP430G2755 ausführt, erweitert um einen Textmodus⁷ in SVGA 800x600 Auflösung, einem schnellen Terminal über USB-CDC, vielen IO-Leitungen und noch viel Platz für eigene kreative Erweiterungen!

Matthias Koch

<https://codeberg.org/Mecrisp/FemtoMSP430>

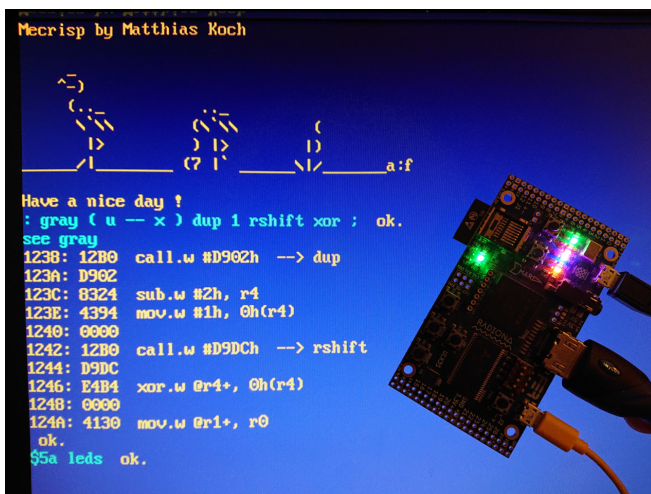


Foto: Matthias Koch

Abbildung 1: Konsolenausgabe (li), ULX3S-FPGA-Board (re)

ULX3S

“The *ULX3S* is a fully open source, compact, robust and affordable FPGA board equipped with

a balanced spectrum of extra components and expansions. Although primarily developed as a teaching tool for mastering the principles of digital circuit design, a wide selection of useful features and interfaces enables it to serve as a broad spectrum module for installation in complex devices ... FPGA: Lattice ECP5 LFE5U-85F-6BG381C (84K LUT) ...”

Quelle: <https://radiona.org/ulx3s/>

mk

Dusk OS — 32-bit Forth OS w/ C compiler

“ ... and big brother to *Collapse OS*. Its primary purpose is to be maximally useful during the first stage of civilizational collapse, that is, when we can't produce modern computers anymore but that there's still many modern computers around.

It does so by aggressively prioritizing simplicity at the cost of unorthodox constraints ... To read more about why this OS exists, see its website.”
[Quelle: <https://git.sr.ht/~vdupras/duskos>]

Lieber Erich Wälde, danke für den Hinweis.

mk

<https://duskos.org/>

Vintage Computerfestival Europa 2026

Die VCFe 25.0⁸ steigt nächstes Jahr am *1. bis 3. Mai 2026* ... Schwerpunktthema: *50 Jahre Billigheimer*.

1975, vor jetzt 50 Jahren, wurde ein Preisbrecher vorgestellt, der die junge Mikroprozessorwelt kräftig aufmischen sollte, die 6502. Als vereinfachte 6800 für weniger als 10% des Preises der 6800 war die 6502 ein idealer Kandidat für die zweite Generation Mikrocomputer und die ersten Heimcomputer. Namen wie Commodore PET, Apple II und Atari 800 wirken bis heute fort. Nicht zu vergessen, Ataris VCS, das für Jahre den Konsolenmarkt bestimmte ...

mk

<https://www.vcfe.org/D/>

⁵ FemtoRV32-Quark — ein RISC-V in 400 Zeilen Verilog — Matthias Koch und Bruno Levy, Heft 4d2021-02

⁶ Vorgestellt im Heft 4d2013-01

⁷ Ein persönlicher Textmodus als Forth-Terminal — Matthias Koch, Heft 4d2025-02

⁸ Danke für den Hinweis, lieber Rafael.

Zensus

Michael Kalus

Wir hatten dieses Jahr in Burladingen (endlich) wieder eine leibhaftige Zusammenkunft und auch der Mitgliedsbeitrag 2025 wurde im November (endlich) wieder erhoben. Und die meisten angeschriebenen Mitglieder haben ihren Tribut inzwischen auch entrichtet — herzlichen Dank dafür. Jeder zählt — weiter so!

Wie viele sind wir in der Forth–Gesellschaft?

Noch Anfang des Jahres war gar nicht klar, wer eigentlich dabei ist. Steht doch in unserer Satzung, dass es keiner formellen Kündigung der Mitgliedschaft bedarf. Jahresbeitrag nicht bezahlen reicht, um nicht mehr im Verein zu sein. Nun, die Gründungsmitglieder des Vereins wollten „keep it simple“ in der Satzung haben!

In der Forth–Gesellschaft (FG) gab es die Möglichkeit, sich trotz Lockdown bei Videokonferenzen zu sehen und in Chatgruppen mitzumachen. Das wurde erfreulich oft genutzt, war ein gewisser Ersatz für die Zusammenkünfte bei den Jahrestagungen, örtlichen Gruppen, größeren Kongressen wie der EuroForth und auf Messen. Manch einem war es sogar lieber so, entfiel doch die weite und kostspielige Reise. So gab es tatsächlich keine aktiven Austritte aus der FG im Lockdown. Die Beiträge wurden wie immer erhoben, ca. 50% als Rechnung, der Rest per SEPA–Lastschrift.

Doch dann kam uns die Postbank (PB) in die Quere, bei der die FG all die Jahre ihr Konto hatte. Seit der Übernahme durch die Deutsche Bank mag man bei der PB Vereine wohl nicht mehr. Die SEPA–Lastschriften wurden nicht mehr ausgeführt und das Vereinskonto gekündigt.

Eine vereinsfreundliche Bank war schnell gefunden: Die Deutsche Skatbank. Aber wegen des *Transparenzregisters* von 2021 müssen GmbHs, Vereine mit wirtschaftlicher Tätigkeit, Stiftungen usw. ihre „Kontrollpersonen“ dort öffentlich machen. Nichteintragung kann mit Bußgeldern bis zu 10% des Jahresumsatzes geahndet werden. Das betrifft uns, denn die FG hat einen Zweckbetrieb: das Forth–Magazin. Also musste die FG erstmal deutlich machen, weder Geldwäsche noch Terrorismusfinanzierung oder organisierte Kriminalität zu betreiben oder als „Briefkasten“ oder zur Verschleierung von Eigentumsverhältnissen missbraucht zu werden — ein Haufen Papierkram für das FG–Büro! So musste sogar vorübergehend das (bescheidene) Vereinsvermögen¹ auf einem Treuhandkonto eines Mitglieds geparkt werden.

Mitglieder Rally²

Das FG–Büro — in Ehrenarbeit und als Freizeitbeschäftigung — war da nicht mehr in der Lage, hinterherzukommen. So gab es für etwa die Hälfte unserer Mitglieder keine Rechnung in 2024.

¹ Siehe Protokolle der MVs

² Rally: Die Strecke besteht aus vielen Prüfungen . . .

³ Die Vier sind der Redaktion namentlich bekannt. :)

Eigentlich wäre der Mitgliederbestand da schlagartig um 50% geschrumpft, und ohne Rechnungsausgang 2025 wären es dann fast 100% geworden. Nur eine ganz kleine Schar Aufrechter hat im Januar 2025 *von sich aus einfach so* ihren Vereinsbeitrag überwiesen — meine Hochachtung, ihr Lieben.³

Im Oktober konnte aber endlich, nach all dem Papierkram, der Vertrag mit der Deutschen Skatbank abgeschlossen und im November das Treuhandkonto dorthin überführt werden — CARSTEN STROTMANN und BERND PAYSAN sei Dank!

Dann wurde die alte Mitgliederliste aufgearbeitet und der Status abgeschätzt, wer wohl noch Mitglied sein möchte. GERALD WODNI gebührt hier unser aller Dank. Er hat es in einer Nachtaktion geschafft, diese Liste in (s)ein Unternehmensnetzwerk „der nächsten Generation“, ins *TeamWorks*, zu bringen und damit die Rechnungen zu erstellen und die Zahlungen zu managen. Den Rechnungsversand hab ich dann gemacht.

Das hat jetzt online im Team gut geklappt und das Forth–Büro ist damit aus der Einzelkämpferecke erfolgreich raus.

Dabei wurde mir sehr deutlich, welch große Leistung CARSTEN STROTMANN und davor EWALD & ANDREA RIEGER im FG–Büro für unseren Verein geleistet haben, still im Hintergrund und ganz ohne Getöse und so ganz allein.

So, nun sind 90 Rechnungen an die uns bekannten Anschriften rausgegangen. Und bisher kamen nur zwei davon unzustellbar zurück, 75% der Rechnungen sind schon bezahlt worden und der Rest kommt hoffentlich bis Weihnachten 2025 noch herein. Und nur zwei Personen haben bei dieser Gelegenheit ihren Vereinsaustritt mitgeteilt.

secretary@forth-ev.de

Die Post–Adressen stimmten wohl größtenteils noch, doch bleibt es unklar, wie man sich eigentlich gegenseitig ohne die Schneckenpost erreichen kann. Die E–Mail–Adressen sind doch sehr unvollständig.

Daher meine Bitte: Schreibt mir an secretary@forth-ev.de ein paar nette Zeilen. Ich antworte prompt, versprochen.



Pili Plop — A Bresenham Variation

Albert Nijhof¹

This program coordinates processes that run with different speeds: they'll start at the same time, run smoothly, and stop at the same time.

Robot arm with (e.g.) five stepper motors

The chairman of the Dutch Forth Users group tinkered with a five-motor robot arm (two in the shoulder, one in the elbow and two in the wrist) and asked me to develop an algorithm to make it move smoothly. But I had absolutely no experience with hardware. Therefore, he took care of the lowest level driver routines himself.

The Task

Send a number to one of the steppers. It'll move abruptly to the position corresponding to that number. It is not necessary to know the present stepper position.

Armed with this hardware knowledge I thought up a program, and, it works! After some tweaking of the word `>MOTOR` of course. Because I did not have any hardware myself, I've tested the result with output to the terminal screen (using `.HERE`).

Because no specific knowledge of the arm is used, there is no protection against commanding the arm steppers to impossible positions.

Variations of the Task

It is trivial to choose the number of motors differently. The program coordinates processes that run with different speeds: they start at the same time, run smoothly, and stop at the same time. Other applications are possible. Examples are cartoons of Achilles and the Tortoise, or drawing straight lines with that five-dimensional plotter you have always wanted to build.

The Problem

Say, we move from position `<0 0 0 0 0>` to position `<10 20 13 7 30>`. The last motor rotates over the largest angle: 30. This largest angle (30) is the starting point of the algorithm: we'll distribute the movement of each motor over 30 segments. Per segment a motor either does a single step (PLOP), or it does nothing (PILI).

The last motor does 1 step in each segment (30 times PLOP). This is simple. Let's look at the other motors.

Distance 10: 10 segments 1 step (PLOP) and 20 segments nothing (PILI). Equally distributed this means: PILI PLOP PILI (10 times).

Distance 20 is simple too: PLOP PILI PLOP (10 times).

Distance 13 is more difficult: 13 times PLOP and 17 times PILI. How to distribute this evenly over 30 segments?

The Solution

Fit the motors with a gas tank.

Fuel

Even when standing still, the motor uses 13 liters per segment (the number of PLOP's, the number of steps to complete the wanted angle). When the fuel goes out, exactly 30 liters are tanked (the number of segments, the largest angle).

Reasoning

Per segment 13 liters are used, over 30 segments: $30 * 13$ liters. $30 * 13 = 13 * 30$. How many times do we need to refuel? Right, 13 times. And how many steps were we supposed to make? ... So here's the solution: If we run out of fuel in a segment, do a refuel and perform PLOP. The other (17) segments are PILI. Now remove those fuel tanks and just remember the reasoning, because that's what we'll use in the code below.

References and Links

<https://wiki.forth-ev.de/doku.php/en:pfw:piliplop>

<https://home.hccnet.nl/anj/ec/ec207a.html>

See also:

<https://home.hccnet.nl/anj/index.html>

<https://home.hccnet.nl/anj/no-ide-nl.pdf>

<https://de.wikipedia.org/wiki/Bresenham-Algorithmus>

Note

The *Bresenham algorithm*, named after JACK BRESENHAM, a programmer at IBM in 1962, is used in computer graphics to draw straight lines on raster displays. It minimizes rounding errors that arise from the discretization of continuous coordinates. The algorithm is easy to implement, requiring only simple addition of integers, thus eliminating the need for multiplication, division, or floating-point operations. This makes it fast and capable of running on even the smallest microcontrollers, making

¹Original was in Dutch. Translation: Marcel Hendrix. Reproduction permission was granted by the author.

it timeless because of its fundamental nature. The principle is applicable to all kinds of iteration problems, as can be seen here.

The following code was written by the author 25 years ago.

Listing

```

47
48 : SEGMENT ( - )
49 #MOTORS
50 0 DO I TANK @ I USAGE @
51 2DUP < \ Fuel shortage?
52 IF #SEGMENTS - \ Refuel
53 I STEP @ I WHERE +! \ New motor position
54 I WHERE @ I >MOTOR \ P-L-O-P
55 THEN
56 - I TANK !
57 LOOP ;
58
59 : GOTO ( -- ) \ The target should be fixed already.
60 PREPARE
61 #SEGMENTS
62 0 ?DO .HERE SEGMENT
63 KEY? IF KEY DROP LEAVE THEN
64 WAIT MS
65 LOOP .HERE ;
66
67 : DOIT ( targetpositions -- ) TARGET GOTO ;
68
69 \ ----- example -----
70
71 10 20 13 7 30 DOIT [rtn]
72 0 0 0 0 0
73 0 1 0 0 1
74 1 1 1 0 2
75 1 2 1 1 3
76 1 3 2 1 4
77 2 3 2 1 5
78 2 4 3 1 6
79 2 5 3 2 7
80 3 5 3 2 8
81 3 6 4 2 9
82 3 7 4 2 10
83 4 7 5 3 11
84 4 8 5 3 12
85 4 9 6 3 13
86 5 9 6 3 14
87 5 10 6 3 15
88 5 11 7 4 16
89 6 11 7 4 17
90 6 12 8 4 18
91 6 13 8 4 19
92 7 13 9 5 20
93 7 14 9 5 21
94 7 15 10 5 22
95 8 15 10 5 23
96 8 16 10 6 24
97 8 17 11 6 25
98 9 17 11 6 26
99 9 18 12 6 27
100 9 19 12 7 28
101 10 19 13 7 29
102 10 20 13 7 30 ok
103
1
2 \ Forth file: ec207.frt
3 \ ROBOT ARM (Albert Nijhof)
4 FORTH DEFINITIONS DECIMAL
5 : VARIABLES
6 CREATE ( n -- ) CELLS ALLOT
7 DOES> ( index body -- adres ) SWAP CELLS + ;
8
9 5 DUP CONSTANT #MOTORS
10 DUP VARIABLES WHERE \ List of motor positions
11 DUP VARIABLES THERE \ List of target positions
12 DUP VARIABLES STEP \ -1 or +1 (directions)
13 DUP VARIABLES TANK \ Amount of fuel
14 DUP VARIABLES USAGE \ fuel used per segment
15 CELLS 0 WHERE SWAP 0 FILL
16
17 20 VALUE WAIT \ Physical motor limitation
18 0 VALUE #SEGMENTS \ Longest segment
19
20 : TARGET ( m0 m1 m2 m3 m4 -- ) \ Set target
21 #MOTORS BEGIN 1-
22 TUCK THERE !
23 ?DUP 0=
24 UNTIL ;
25
26 : PREPARE ( -- )
27 0 \ Longest segment
28 #MOTORS \ Per motor:
29 0 DO I THERE @ I WHERE @
30 2DUP <
31 DUP 2* 1+ I STEP ! \ 1 of -1, direction
32 IF SWAP THEN -
33 DUP I USAGE ! \ Distance
34 MAX \ Largest distance?
35 LOOP
36 DUP TO #SEGMENTS
37 2/ #MOTORS
38 0 DO DUP I TANK ! \ Tanks half full
39 LOOP DROP ;
40
41 \ To use on-screen without hardware
42 : >MOTOR ( new-position motor# -- ) 2DROP ;
43
44 \ To put motor information on screen
45 : .HERE ( -- )
46 CR #MOTORS 0 DO I WHERE @ 4 .R LOOP SPACE ;

```

RV003eForth — eForth für RISC-V

Wolfgang Strauß

Als ich im Jahr 2018 die Forth-Welt entdeckte, dauerte es nicht lange, bis ich auf die Meinung stieß, dass man auf jeden Fall in seinem (Forth-)Leben irgendwann ein eigenes Forth schreiben müsse. Ich habe das damals für mich kategorisch ausgeschlossen. Ich sah mich als Forth-Nutzer, speziell im Bereich Mikrocontroller, und wollte mich auf die Entwicklung von Anwendungen und Lehr-/Schulungs-Material konzentrieren. Keine fünf Jahre später ist es dann doch passiert. Na, ja, zumindest habe ich mich in die Richtung bewegt. Nicht wirklich ein „neues“ Forth von Grund auf selbst entwickelt, sondern ein bestehendes auf eine neue Architektur portiert. Wobei der Begriff „portiert“ es nicht wirklich umfänglich beschreibt. Es steckt schon etwas mehr dahinter, als stumpf die Assembler-Befehle von einer Architektur auf eine andere zu „übertragen“. Von meiner Reise und dem Drumherum möchte ich im Folgenden berichten.

Vorgeschichte — wie es zu dem Wunsch kam, ein Forth zu portieren

Im Oktober 2022 schwappte eine Meldungswelle durchs Internet der Maker-Szene. Der chinesische Hersteller *WCH Qinheng Microelectronics* hatte einen Chip auf den Markt gebracht, der angenehme technische Eigenschaften mit einem Feature verband, dem kein Elektronik-Blog widerstehen konnte: In moderaten Stückzahlen sollte der Baustein gerade mal 10 Cents kosten.

Zuvor waren zwar schon vereinzelt billige Mikrocontroller aus China im Westen bekannt geworden, die aber immer die eine oder andere Unzulänglichkeit aufwiesen. Entweder waren sie schwer bis gar nicht erhältlich oder die Dokumentation nur in Chinesisch vorhanden oder es waren OTPs¹ und so weiter.

Doch nun sah es so aus, als ob dieser neue Chip mit der Typenbezeichnung *CH32V003* alle wichtigen Forderungen der Maker-Community erfüllte. Akzeptable Specs, brauchbare englische Dokumentation, freie Entwicklungswerkzeuge und natürlich als Bonus den unschlagbaren Preis obendrauf.

In der Folge erschienen Blog-Artikel, YouTube-Videos und schließlich sogar ein Open-Source-Projekt (*ch32fun*, auf GitHub), mit dem der *CH32V003* in Betrieb genommen werden kann, ohne auch nur ein Bit der Software des chinesischen Herstellers anrühren zu müssen. Unglaublich, welche Energien da freigesetzt wurden.

Ich wurde aufmerksam auf das Bauteil, als *DAVE JONES* auf seinem YouTube-Kanal *EEVblog* darüber berichtete und vorführte, wie leicht die Chips handzuhaben sind. Für mich war klar, dass ich mich mit dem *CH32V003* beschäftigen wollte. RISC-V fasziniert mich und es war für mich klar, dass irgendwer auf der Welt in Kürze ein Forth für diesen „10-Cent-Mikrocontroller“ entwickeln würde.

Also wartete ich geduldig und schaute regelmäßig in die einschlägigen Kanäle, ob sich in Sachen Forth auf dem *CH32V003* etwas tut. Aber, völlig unbegreiflich für mich, es tat sich nichts in Richtung Forth. Dann war irgendwann die Welle vorbei und es wurde ruhig um den Billigheimer aus China. Noch hier und da ein YouTube-Video von

diversen Makern, die abgefahrene Sachen mit dem Chip anstellten, aber nichts in Richtung Forth.

Also fragte ich in Forth-Kreisen herum, ob nicht jemand ein Forth für diesen tollen Baustein schreiben wolle. Es wollte keiner. Normalerweise wäre das das Ende für Forth auf dem *CH32V003* gewesen und auch ich hätte mich wieder mit anderen interessanten Dingen rund um das Thema Forth beschäftigt.

Es gab da nur ein Problem: Mich ließ der Gedanke nicht los, wie wunderbar es wäre, ein kleines Forth auf diesem Controller zu haben. Wäre es überhaupt möglich, einen akzeptablen Funktionsumfang in einem 32-Bit-Forth in 16 KB Flash unterzubringen und noch ausreichend Platz für das Anwenderprogramm übrig zu lassen?

Und so kam es, wie es kommen musste. Ich entschloss mich, die Sache selbst in die Hand zu nehmen und es Realität werden zu lassen: *Forth auf dem CH32V003*.

Braucht die Welt noch ein Forth?

Wer weiß schon genau, was die Welt wirklich braucht? Zu meiner Entschuldigung kann ich anführen, dass die Welt kein neues Forth bekommen hat, sondern eine weitere Variante des gut eingeführten und durchaus beliebten eForths von *CHEN-HANSON TING*.

Ich kann mir auch gut vorstellen, dass Dr. Ting, hätte er zu Lebzeiten noch die Möglichkeit gehabt, selbst auf die Idee gekommen wäre, sein eForth auf RISC-V zu bringen. Man kann auch nie wissen, welche Impulse so eine Entwicklung in die Forth-Gemeinschaft bringt. Neue Impulse kann Forth immer gut gebrauchen, finde ich.

Warum Tings eForth für den MSP430 als Vorlage?

Als Vorlage für ein Forth auf dem *CH32V003* eignet sich *Dr. Tings MSP430eForth* wunderbar. Der *MSP430* (speziell der *MSP430G2553*) und der *CH32V003* ähneln sich. Beides sind RISC-Architekturen, aber was wichtiger ist: Beide haben ungefähr den gleichen Speicherausbau und vergleichbare Peripherie.

¹One-Time Programmable, nur ein mal programmierbar, nicht wieder löschar

Ein nicht zu unterschätzender Faktor ist, dass das MSP430eForth sehr gut dokumentiert ist und in der Praxis für reale Projekte eingesetzt wird. MICHAEL KALUS hat das MSP430eForth auf einen Open-Source-Assembler übersetzt (naken_asm), MANFRED MAHLOW hat zahlreiche Verbesserungen in Punkto Betriebssicherheit und Komfort beigesteuert und Fehler gefunden und beseitigt. Darauf kann man gut aufbauen.

Als weiteres Argument darf natürlich die Kompaktheit des MSP430eForth nicht unerwähnt bleiben. Es „wiegt“ nur etwas über 4200 Bytes. Ich kenne kein praxistaugliches Forth, das weniger Flash belegt.

Die Hoffnung war, dass mein *RV003eForth* mit 8 KB auskommen würde. Dann hätte der Anwender die restlichen 8 KB für seine Zwecke zur Verfügung. Aber ich tappte völlig im Dunklen, was den realen Speicherbedarf eines RISCV Forths angeht. Die einzige Zahl, die ich kannte, waren die knapp 20 KB vom Mecrisp Quintus. Aber natürlich sind eForth und Mecrisp zwei verschiedene Paar Schuhe.²

Es war für mich übrigens von Anfang an ein treibender Faktor, herauszufinden, wie das RV003eForth im Vergleich zum MSP430eForth abschneiden würde. Sowohl vom Speicherbedarf als auch von der Geschwindigkeit. Ich sollte eine Überraschung erleben ...

RV003eForth — erste Schritte

Wie geht man so ein Projekt an?

Zunächst ist da die Einarbeitungsphase. Studieren der Vorlage (MSP430eForth + PDF „Zen“ von Ting). Immer schon mit dem Gedanken im Hintergrund, wie die einzelnen Features auf dem neuen Prozessor umgesetzt werden können.

Dann das „Erobern“ des neuen Chips. Natürlich alles erst mal in Assembler. Ein lauffähiges Forth ist noch in weiter Ferne. Grundfunktionalität herstellen wie Einstellen der gewünschten Taktfrequenz, Initialisieren der seriellen Peripherie zur Kommunikation mit dem Host (PC), Zeichenein- und ausgabe, Stringausgabe, Zahlenausgabe (zunächst nur in Hexadezimal). Kleine Hilfsprogramme, wie z. B. ein Memory-Dump. Damit lässt sich schon ordentlich arbeiten und debuggen.

Der nächste Schritt war aufwändig: den Flash-Controller unter Kontrolle bringen. Flash beschreiben und seitenweise löschen. Was man da alles beachten und einstellen muss, bis man so ein lumpiges Byte ins Flash geschrieben bekommt, ist schon erstaunlich. Da müssen z. B. mehrstufige Sicherheitssperren entriegelt und nach dem Schreiben wieder verriegelt werden. Aber ich will mich gar nicht beschweren. Ich kenne noch die Zeit, wo man für jede Änderung des Programmcodes den Speicherbaustein (EPROM) aus der Platine entfernen, mit UV-Licht löschen, in einem Programmiergerät beschreiben und abschließend wieder in die Platine einsetzen musste. Nur

²eForth: Direct-Threaded. Mecrisp: Subroutine-Threaded.

um dann festzustellen, dass das Programm nicht wie gewünscht läuft und der ganze Vorgang wiederholt werden muss. Notiz an mich selbst: Früher war eben doch nicht alles besser.

Den CH32V003 unter Kontrolle zu bringen und die Kommunikation mit dem PC herzustellen, war spannend und im Ergebnis sehr befriedigend. Ich bewegte mich auf bekanntem Terrain. Was mich nun erwartete, war Neuland.

Ein Forth erwacht

Was ist der nächste logische Schritt?

Ganz klar, der innere Interpreter (auch virtuelle Forth-Maschine genannt) muss laufen. Und zwar flüssig, denn hier entscheidet sich, wie performant das Forth später ist. Doch halt! Da ist auch noch die andere „Sache“. Es gilt, speicherschonend zu programmieren, denn 16 KB Flash sind nicht gerade als üppig zu bezeichnen. Bei einem Prozessor mit 32-Bit-Opcodes entspricht das gerade mal 4096 Befehlen, dann ist der Speicher aufgebraucht. Und zwar komplett. Die Entscheidung ist schnell getroffen. Priorität: Das Forth muss klein bleiben.

Im zweiten Schritt kann man dann schauen, wo noch etwas Performanz herausgeholt werden kann. Nächste Entscheidung: In welchem Modus soll der innere Interpreter denn laufen? Subroutine-, Indirect- oder doch Direct-Threaded (STC, ITC, DTC)? Subroutine-Threaded habe ich gleich verworfen. Es hat zwar den charmanten Vorteil für einen Forth-Einsteiger, einfach zu verstehen zu sein, da der innere Interpreter komplett durch die Hardware der CPU abgebildet wird. Die mangelnde Speichereffizienz disqualifiziert es jedoch aus meiner Sicht. Direct- oder Indirect-Threaded, das ist hier also die Frage.

Tings MSP430eForth ist ein Direct-Threader. Aber interessanterweise sind frühere Versionen seines eForths Indirect-Threader. Hier zeigt sich für mich deutlich, dass es gar nicht so einfach ist (auch für jemanden mit erheblich detaillierteren Kenntnissen als ich sie aufweisen kann), hier die optimale Auswahl zu treffen. Ting hat seine endgültige Wahl nach ausgiebigen Benchmarks der beiden Systeme getroffen. Meine eForth-Variante ist ein Direct-Threaded-Forth, weil sie speicherschonender ist. Primitives (also Worte, die direkt Maschinenbefehle abarbeiten) haben bei DTC keinen einleitenden Vektor, der auf den Codeteil zeigt. Da Dr. Ting das MSP430eForth mit ungewöhnlich vielen Primitives ausgestattet hat, liegt hier ein nicht zu unterschätzendes Sparpotential.

Dank an Klaus Kohl-Schöpe, der mich in dieser Phase des Projekts mit Informationen und Ideen versorgt hat. Sogar der Prototyp eines inneren Interpreters eines ITC-Forths war dabei. Dort habe ich zum ersten mal gesehen und begriffen, wie elegant und kompakt Forth im Inneren werkelt.

Also auf ans Werk und einen DTC-Interpreter geschrieben. RV003eForth ist ein 32-Bit-Forth. Heißt, eine „Zelle“ hat 32 Bits, Stackeinträge, CONSTANTS und VARIABLES haben immer Zell-Größe. Aber: Die Token-Listen der Compound-Worte habe ich 16-bittig ausgeführt.

Mit 16 Bits kann man einen Adressraum von 64 KB überstreichen; mehr als genug für die überschaubaren 16 KB des vorgesehenen Mikrocontrollers. Literals (also auf den Datenstack zu pushende Konstanten, die in Compound-Worte hineincompiliert werden), sind 16- oder 32-bittig. Die Wahl erfolgt automatisch, je nach Platzbedarf. Werden diese Literals später bei Ausführung des Wortes auf den Datenstack gepusht, sind sie vorher auf 32 Bit „aufgeblasen“ worden, falls sie im 16-Bit-Format vorlagen.

Das verkompliziert die Logik etwas, verspricht jedoch eine erhöhte Code-Dichte. Mal sehen, ob die Praxis dies später bestätigt. Das hängt natürlich maßgeblich von dem Literal-Mix in den Anwender-Programmen ab.

Zu meiner Überraschung lief der innere Interpreter sofort ohne Fehler. Der erste Test war die Abarbeitung einer einfachen, verschachtelten Hierarchie von Compound- und Primitive-Worthülsen, um zu sehen, ob der Return-Stack sich ordnungsgemäß verhielt. Tat er. Kein Unter- oder Überlaufen. Völlig unverdächtig.

Ab wann ist ein Forth ein Forth? Gibt es eine Definition? Reicht ein laufender innerer Interpreter? Vermutlich ist das noch ein bisschen wenig Forth, vielleicht kann man es eine Forth-Keimzelle³ nennen.

Tiefschlaf

Nach dem schnellen Erfolg des laufenden inneren Interpreters machte ich mich ans Werk, die Directory-Struktur zu übersetzen. Die Vorlage dazu lieferte der von MICHEAL KALUS und MANFRED MAHLOW überarbeitete Assembler-Quelltext des originalen MSP430eForths von Dr. Ting. Stolze 2229 Zeilen hat die Datei. Darin enthalten sind über 60 Primitives.

Das bedeutet, dass diese Worte in MSP430-Maschinencode ausgeführt sind und von Hand in RISC-V Maschinencode „übertragen“ werden müssen. Übertragen ist dabei nicht wörtlich zu nehmen, denn es ist ja klar, dass nicht für jeden MSP430-Befehl ein entsprechender RISC-V-Befehl existiert. Da muss man zum Teil „um die Ecke“ denken.

Ich kann MATTHIAS KOCH nicht genug danken für sein geniales *Mecrisp Quintus*. In den Quintus-Quellen zu stöbern, ist wie ein gutes Buch zu lesen. Unterhaltung pur für Nerds. RISC-V Programmierung vom Feinsten. Da hat jemand RISC-V nicht nur verstanden, sondern verinnerlicht. Ich gestehe gerne, dass ich mir dort den einen oder anderen Programmier-Kniff „ausgeborgt“ habe.

Nachdem es anfangs gut lief, wurde es zunehmend anstrengender. Anstrengend im Sinne von „am Ball bleiben“ und die Fleißaufgabe der Übersetzung zu Ende zu bringen.

³ Engl.: kernel = der Kern des Betriebssystems :)

Da gab es keine Belohnung mehr auf dem Weg. Die würde erst am Ende gewährt, wenn die Portierung fertiggestellt war: der Willkommensprompt von Forth am Bildschirm und die Möglichkeit des interaktiven Arbeitens.

Die Pausen zwischen den Arbeiten wurden immer länger und irgendwann schlief die Sache dann komplett ein. Andere Dinge schoben sich in den Vordergrund. Monat um Monat ging ins Land ...

Mammut-Akt

Nach einer Pause von zwei Jahren fand ich zurück ins Projekt. Ich setzte mir eine Deadline. Bis zur nächsten Tagung im April 2026 in Bochum wollte ich das Forth betriebsbereit haben.

Die Motivation war wieder da und ich startete durch. Innerhalb von 2 Wochen schloss ich die Übersetzungsarbeiten ab. Ich hielt mich strikt an den gemachten Plan und hakte fleißig jede Teilaufgabe ab. Das Ende kam näher und schließlich war der Tag da, auf den ich so lange hingearbeitet hatte. Würde ich meine Belohnung erhalten? Würde mich Forth mit der Willkommensmeldung begrüßen und meine Eingaben entgegennehmen und abarbeiten? Oder warteten endlose und zermürbende Debug-Sitzungen auf mich?

Ich flashte das Forth in den Mikrocontroller, startete das serielle Terminal und drückte die Reset-Taste — die Forth-Einschaltmeldung erschien. Ich konnte es kaum glauben. Aber es gab keinen Zweifel. Da stand in grünen Buchstaben auf schwarzem Grund die Zeichenfolge „RV003eForth“.

Ich drückte die Taste 'w' und siehe da, ein w erschien auf dem Bildschirm. Das ließ sich doch gut an. Ich komplettierte zu words und drückte Enter. Jetzt kam eine Fehlermeldung „words?“. Oha. Das war nicht gut. Ich wusste, dass das Wort im Dictionary vorhanden war. Ich probierte .s . Die gleiche Fehlermeldung. Ich probierte herum und irgendwann gab ich eher aus Zufall .S ein. Jetzt kam keine Fehlermeldung mehr, sondern eForth gab den (leeren) Stackinhalt aus.

Tja, ich war einem typischen Anfängerfehler zum Opfer gefallen. eForth ist nämlich *case-sensitiv*. Wer lässt sich so etwas einfallen? Kaum gab man WORDS ein, spuckte eForth brav die Liste der Worte im Dictionary aus.

Ich legte eine Wunschliste an. Erster Eintrag: „RV003eForth case-insensitiv machen.“

Es läuft noch nicht rund

Bevor ich jedoch dazu kam, meinen Wunschzettel abzuarbeiten, musste noch ein hartnäckiger Fehler beseitigt werden. Hin und wieder stürzte eForth scheinbar zufällig ab, nur um ein anderes Mal klaglos durchzulaufen.

Das sind die Fehler, an denen man verzweifeln kann. Ich fand erst kein Muster des Fehlers und wusste nicht, wo

ich suchen sollte. Dann kam mir ein Verdacht. RISC-V Prozessoren können in unterschiedlichen Breiten auf Daten zugreifen. 32-Bit-Prozessoren haben 8-, 16- und 32-Bit-breite Zugriffe im Angebot. 8-Bit-Zugriffe sind problemlos, es werden keine besonderen Anforderungen an die Adressen gestellt. Anders sieht es bei 16- und 32-Bit-Zugriffen aus. Da stellt es der Standard jedem Chiphersteller frei, wie „freizügig“ der Zugriff implementiert ist. 16-Bit-Zugriffe sind oft nur auf geraden Adressen erlaubt (das niederwertigste Bit der Adresse muss Null sein). Bei 32-Bit-Zugriffen gibt es 2 gängige Varianten. Variante A: Es reicht, wenn die Adresse gerade ist. Variante B: Die Adresse muss glatt durch 4 teilbar sein (die zwei niederwertigsten Bits sind Null).

Ich bin blauäugig⁴ davon ausgegangen, dass der Hersteller Variante A beim Datenzugriff implementiert hat, weil das beim Befehlszugriff auch funktioniert (der Chip unterstützt 16- und 32-Bit-Befehle, die frei gemischt werden dürfen). Pustekuchen. Der Hersteller hat Variante B gewählt. Und das hat unangenehme Konsequenzen. Versucht man z. B., einen 32-Bit-Wert an einer Adresse zu lesen, die nicht glatt durch 4 teilbar ist, kracht es. Der normale Programmablauf wird unterbrochen und der Prozessor springt zu einer Ausnahmebehandlungsroutine.

Von außen sieht es so aus, als sei das System eingefroren, weil ich mir die Sache einfach gemacht habe und die Behandlungsroutine aus einer Endlosschleife besteht. Das Problem ist jetzt, dass im Dictionary zwar überwiegend 16-Bit-Daten (z. B. Token, 16-Bit-Literals) stehen, die natürlich und unproblematisch mit 16-Bit-Zugriffen gelesen werden, aber leider auch hier und da ein 32-Bit-Wert (32-Bit-Literale, Konstantenwerte und Adressen von Variablen). Diese Objekte habe ich mit 32-Bit-Zugriffen gelesen. Wenn die Daten zufällig auf einer durch 4 teilbaren Adresse lagen, lief der Zugriff durch. Lagen sie „nur“ auf einer geraden Adresse, fror das System ein.

Als ich den Fehler eindeutig identifiziert hatte, lag die Lösung auf der Hand: keine 32-Bit-Zugriffe im Dictionary-Speicherbereich. 32-Bit-Zugriffe werden nun unter Verwendung von 16-Bit-Befehlen „emuliert“:

```
lhu  tos, 2(ip)    # load upper 16 bits
slli  tos, tos, 16 # shift left 16 positions
lhu  aa, 0(ip)    # load lower 16 bits
or   tos, tos, aa # merge into 32 bit value
```

Das kostet ein paar Takte zur Laufzeit, hat aber keine Nachteile beim Speicherbedarf, da der Code Teil des inneren Interpreters ist. Die Alternative wäre, dafür zu sorgen, dass 32-Bit-Daten im Dictionary garantiert auf durch 4 teilbaren Adressen liegen, indem man bei Bedarf 2 „Stopfbytes“ eincompiliert. Das ist mir zu teuer, denn der Flash-Speicher ist knapp bemessen.

Nachdem dieses schwer zu debuggende Problem erledigt war, lief das Forth schon ganz ordentlich. Einige Worte waren noch mit Fehlern behaftet, aber die wunderbare

⁴ Meine Augenfarbe ist tatsächlich blau, aber ich meine hier etwas anderes :-)

Modularität von Forth sorgt dafür, dass die meisten Worte isoliert getestet und schnell von Bugs befreit werden können.

Wünsch Dir was

Ich vermute, dass bei jedem Programmierer, wenn er sich nur lange und intensiv genug mit einer Programmiersprache beschäftigt, Änderungs- und/oder Erweiterungswünsche aufkommen. Bei den meisten Sprachen bleibt es bei dem Wunsch, denn da ist keine Schraube vorhanden, an der man drehen könnte.

Forth aber besteht fast vollständig aus Schrauben, um mal bei diesem Bild zu bleiben. Einige Schrauben haben Abdeckkappen, die sich jedoch leicht lösen lassen, wenn man nur fest genug drückt. Macht es Sinn, an jeder Schraube zu drehen, nur weil man es kann? Wahrscheinlich nicht, denn die meisten Schrauben haben genau das richtige Anzugsdrehmoment. Darf man an einigen ausgesuchten Schrauben drehen, weil sie für den persönlichen Gebrauch zu fest oder zu lose sitzen? Bestimmt.

Es folgt eine Sammlung der Änderungen/Erweiterungen, die ich mir erlaubt habe, vorzunehmen. Es sind auch immer die Kosten in Bytes angegeben. So könnt ihr selbst entscheiden, ob euch das Feature den Speicherbedarf wert wäre oder ob ihr lieber den Speicher hättet.

DUMP

Eine kleine Menge Tools ist standardmäßig in eForth eingebaut. DUMP ist dabei. Das zeigt die Nützlichkeit, die Chen-Hanson Ting diesem Speicherabbilddrucker zuschrieb. Während es in anderen Forthen nachgeladen werden muss, steht es in eForth immer zur Verfügung.

Kosten für DUMP: +68 Bytes

Tings DUMP hat aus meiner Sicht eine kleine Schwäche. Es zeigt die Adressen und Speicherinhalte immer in der aktuellen Zahlenbasis an. Arbeitet man gerne im Dezimalmodus (so wie ich), ist die Anzeige zumindest ungewohnt, wenn nicht sogar unbrauchbar, da Adressen (und auch Inhalte) in der Literatur (Datenblätter) gerne in Hexadezimal angegeben sind. Klar, man kann vor der Benutzung von DUMP mit HEX umschalten und danach wieder mit DECIMAL zurück, aber will man das? Abhilfe: das Wort selbst die Umschaltung vornehmen lassen.

Kosten für HEX-Automatik: +14 Bytes

Zahleneingabe

NUMBER? hat beim eForth die Möglichkeit der Eingabe von Hexadezimalzahlen mit dem \$-Präfix fest eingebaut. Auch das ein Detail, welches zeigt, das Ting sein Forth an der Praxis orientiert hat. Leider fehlt das Präfix '%' für Binärzahlunterstützung. Ich habe es nachgerüstet.

Kosten für %-Binärzahlen: +26 Bytes

Weil ich gerade dabei war, habe ich auch das #-Präfix aus dem Forth-Standard 200x eingebaut. Damit kann man sicherstellen, dass Forth eine Zahl als Dezimalzahl interpretiert, egal, welche Basis gerade eingestellt ist.

Kosten für #-Dezimalzahlen: +20 Bytes

Der Forth-Standard sieht Zeichenkonstanten in der Form 'x' vor (ein Zeichen, eingeschlossen in einfache Anführungszeichen). Da eForth kein [CHAR] anbietet, habe ich das Feature als Alternative aufgenommen.

Kosten für Zeichenkonstanten: +58 Bytes

Nun zu einem meiner Lieblingsthemen. Einige Programmiersprachen unterstützen den Unterstrich ('_') als Separator zwischen Ziffern bei der Zahleneingabe, um die Lesbarkeit zu erhöhen. Oder könnt ihr mit einem Blick sagen, wie viele Nullen die folgende Zahl hat?

48000000

Und jetzt?

48_000_000

Noch ein Beispiel, dieses Mal mit Binärzahlen. Na, welchen Wert hat das Bitfeld [10:7] ?

%1010100100100010

Wie wäre es damit:

%10101_0010_0100010

Tipp- oder Lesefehler bei Zahlen können zu langen Debugnächten führen (alles schon mitgemacht). Den Unterstrich gönne ich mir. Wieso ist der eigentlich noch nicht im Forth-Standard angekommen? Gab es da schon mal ein Proposal?

Kosten für den Zifferseparator: +38 Bytes

Wer HEX sagt, muss auch BINARY sagen

Kurz und schmerzlos: DECIMAL und HEX sind im eForth vorhanden, BINARY fehlt. Das geht natürlich nicht. Habe es hinzugefügt.

Kosten für BINARY: +22 Bytes

Schieberei!

Das MSP430eForth hat keine komfortable Shift-Unterstützung. Kein LSHIFT RSHIFT ARSHIFT. Wird vermutlich daran liegen, dass der MSP430 keinen Barrel-Shifter hat und deshalb pro Befehl immer nur um eine Stelle schieben kann. Das wird teuer zur Laufzeit. RISC-V unterstützt hingegen mehrstufiges Schieben pro Befehl. Sehr angenehm. Ich habe die Worte LSHIFT RSHIFT und ARSHIFT ins Dictionary aufgenommen.

Kosten für LSHIFT RSHIFT ARSHIFT: 3x +20 Bytes

Groß und klein, muss das sein?

Dass eForth auf exakter Groß- und Kleinschreibung besteht, hat mich kalt erwischt. Damit hatte ich noch nicht zu tun. Die Forthe, mit denen ich bis dato gearbeitet habe, sind da nicht so streng. wORDs und WORDs ist für sie das selbe. Ich habe versucht, mich mit der „Eigenart“ von eForth anzufreunden, allein, es will mir nicht gelingen. Ich sehe zwei Möglichkeiten: entweder mich weiter zu ärgern oder selbst Hand anzulegen. Ich wähle Tor zwei und mache eForth *case-insensitiv* und damit für mich wieder sympathisch. Es ist an zwei Stellen eine Änderung nötig. Bei der Directory-Suche und bei der Zifferauswertung der Zahleneingabe. Ich möchte ja schließlich bei Hexadezimalzahlen auch frei sein, entweder „a-f“ oder „A-F“ zu benutzen.

Kosten für Case-Insensitivität: +48 Bytes

Tick-Tack

Immer mal wieder muss in Mikrocontrollerprogrammen eine gewisse Zeit verstreichen, bis es weitergehen kann. Man denke an die LED, die für 300 ms leuchten soll etc. Die einfachste Art, so etwas zu erledigen, ist, den Prozessor in einer Schleife die Zeit „verbrennen“ zu lassen. Das ist nicht besonders elegant, da der Prozessor in dieser Zeit nichts anderes tun kann, als seine Runden zu drehen. Für simple Anwendungen reicht es aber.

Ich habe deshalb das Wort MS in den Forth-Kernel aufgenommen, da ich es dauernd brauche und es so eh nachladen müsste. 300 MS wartet z. B. 300 Millisekunden.

Kosten für MS: +36 Bytes

Neuralyzer

eForth kann nur ins Flash compilieren. So füllt sich bei der interaktiven Softwareentwicklung nach und nach der kostbare Festwertspeicher; auch mit alten Versionen redefinierter Worte. eForth hat kein eingebautes FORGET oder ähnliche Mechanismen anzubieten. Wie wird man diesen Ballast also wieder los? Am radikalsten ist, den Controller einfach komplett zu löschen und mit einem frischen Forth zu bespielen. Das wird aber schnell lästig, da der Vorgang mehrere Schritte umfasst.

Ich habe deshalb das Wort FACTORY-RESET implementiert. Es stellt den Zustand eines frisch geflashten eForthes wieder her und führt einen sauberen Software-Reset aus. Mit abgefallen ist das Wort RESET, welches nur einen Reset einleitet, das Flash aber nicht antastet. Das ist praktisch, da nicht alle Gehäusevarianten des CH32V003 einen Resetpin haben.

Kosten für FACTORY-RESET: +194 Bytes

Anmerkung: In den 194 Bytes ist ein etwa 50-Byte-langer Mitteilungstext enthalten, der den User über die getätigte Aktion informiert. Hier könnten leicht mehrere 10 Bytes eingespart werden, wenn gewünscht.

Gez ma Butta bei die Fische!⁵

Was „wiegt“ denn nun das RV003eForth?

Wie ich ja eingangs schrieb, hoffte ich, mit einem Speicherbedarf von 8 KByte für das RV003eForth auszukommen. Das RV003eForth mit den besprochenen Erweiterungen hat nun eine Größe von etwa 5,5 KByte (Stand 2025-11-28). Nicht schlecht, oder? Rechne ich meine Erweiterungen heraus, so dass ich die beiden eForth-Varianten direkt vergleichen kann, dann legt das 32-bittige RV003eForth nur 850 Bytes mehr auf die Waagschale als die 16-bittige MSP430-Variante.

Man könnte auch sagen: MSP430eForth + 20% = RV003eForth. Für den Anwender bleiben somit etwas mehr als 10,5 KByte für die eigene Anwendung. Damit kann man arbeiten, meint Hubert.

Und wie „rennt“ das RV003eForth?

Ich habe ein paar einfache Benchmarks durchgeführt. Ergebnis: Bei gleicher Taktfrequenz ist das MSP430eForth um den Faktor 3 schneller als RV003eForth. Wie bitte? Wie kann das sein, dass der MSP430 den RISC-V so „aus dem Wasser bläst“? Der MSP430 hat mit `mov @ip+,pc` einen Befehl, der das komplette NEXT abbildet. Besser geht es nicht. Das muss bei RISC-V mit 3 Befehlen „nachgebaut“ werden. Schwacher Trost: Lässt man beide MCUs mit ihrem maximalen Takt antreten (MSP430: 16 MHz, CH32V003: 48 MHz), liegen die beiden eForths gleichauf.

DTC unter der Haube

Primitive Command

```

-----
|      Link Field      |
+-----+
|      Name Field     |
+-----+
| Machine Instruction |
|      :              |
| Machine Instruction |
|  c.jr addr_NEXT_   |
,-----,

```

```

# NEXT sequence
lhu temp,0(ip) # load execution token in temp
c.addi ip,2    # increment ip by token size
c.jr temp     # jump to address in temp

```

Primitives sind Worte, die am Ende der Aufrufhierarchie stehen. Sie starten direkt mit Maschinencode (16- oder 32-bittig) und enden mit einem Sprung zur NEXT-Sequenz. Dieser Sprung ist hier als indirekter Sprung an eine Adresse ausgeführt, die dauerhaft in einem Register hinterlegt ist. Das spart Speicherplatz.

⁵ Geflügeltes Wort im Ruhrgebiet.

Compound Command

```

-----
|      Link Field      |
+-----+
|      Name Field     |
+-----+
| c.jalr addr_DOLST |
|      Address Token  |
|      :              |
|      Address Token  |
|      EXIT Token    |
,-----,

```

```

DOLST:
addi sp,sp,-4 # \ push ip to return stack
sw ip,0(sp)   # /
mv ip,ra     # put addr of first token in ip
# NEXT
lhu temp,0(ip) # load execution token in temp
c.addi ip,2    # increment ip by token size
c.jr temp     # jump to address in temp

```

```

EXIT:
lw ip,0(sp)   # \ pop ip from return stack
addi sp,sp,4  # /
# NEXT
lhu temp,0(ip) # load execution token in temp
c.addi ip,2    # increment ip by token size
c.jr temp     # jump to address in temp

```

Jedes Compound-Wort hat am Anfang einen Aufruf des DOLST-Codes, der die Abarbeitung der Token-Liste einleitet. Die Liste wird mit dem EXIT-Token abgeschlossen. Das Besondere ist, dass die Token aus 16-Bit-Adressen bestehen, obwohl das Forth 32-bittig ist.

Defining Words: CONSTANT, VARIABLE, CREATE

```

-----
|      Link Field      |
+-----+
|      Name Field     |
+-----+
| c.jalr addr_DOCON |
| 32-bit value       |
,-----,

```

```

DOCON:
addi stack,stack,-4 # \ effectively a DUP
sw tos,0(stack)    # /
lhu tos,2(ra)     # load upper 16 bits
slli tos,tos,16   # shift left 16 positions
lhu temp,0(ra)    # load lower 16 bits
or tos,tos,temp   # merge into 32-bit value
# NEXT
lhu temp,0(ip) # load execution token in temp
c.addi ip,2    # increment ip by token size
c.jr temp     # jump to address in temp

```

Der DOCON-Code lädt hier den 32-Bit-Wert auf eine umständlich anmutende Weise mittels zweier 16-Bit-Zugriffe auf den Datenstack, obwohl das theoretisch mit einem 32-Bit-Lesebefehl erledigt wäre. Der Grund ist weiter vorne im Artikel beschrieben.

RAM-Belegung RV003eForth

Der CH32V003 hat 2048 Bytes RAM. Da muss man gut haushalten.

```

.----- . --- ($2000_0800)
| 16 Bytes |
+-----+ RPP ($2000_07F0)
| Return Stack |
|   |||   |
|   vvv   |     ^^^
|         |     240 Bytes
|   ^^^   |     vvv
|   |||   |
| Terminal |
| Input Buffer |
+-----+ TIBB ($2000_0700)
| 16 Bytes |
+-----+ SPP ($2000_06F0)
| Data Stack |
|   |||   |
|   vvv   |     ^^^
|         |     1712 Bytes
|   ^^^   |     vvv
|   |||   |
| Variables & |
|   Data   |
+-----+ DPP ($2000_0040)
| User     |     64
| Variables |     Bytes
'-----' UPP ($2000_0000)

```

Wer schon mal einen Speicherplan eines Forthes gesehen hat, wird hier keine Auffälligkeiten oder besonderen Tricks finden. Returnstack und Terminal-Input-Buffer wachsen aufeinander zu, ebenso wie der Datenstack und der allozierbare Datenbereich sich aufeinander zubewegen. Lässt der Programmierer zu, dass sich die Lücken schließen, wird das Forth einen Hüllenbruch⁶ erleiden.

Rant⁷: Assembler

Mit dem Begriff Assembler ist hier das Programm gemeint, das einen Assembler-Quelltext in Maschinencode wandelt und in eine Datei schreibt.

Man sollte doch meinen, da gäbe es nicht viel zu erzählen oder gar zu bemerken. Ein Assembler arbeitet unauffällig und verrichtet seine Arbeit. Ein Assembler wird für die Erstellung eines eForthes ja auch nur gebraucht, um den Forth-Core in Maschinencode zu übersetzen, damit der in den Mikrocontroller geflasht werden kann. Dann hat

so ein Assembler erst einmal Pause, bis Veränderungen am Core gewünscht werden.

GAS

Ich habe für die ersten Experimente mit dem CH32V003 zunächst den GAS (GNU Assembler) verwendet. Mit diesem Assembler hat MATTHIAS KOCH sein *Mecrisp Quintus* entwickelt. Gut genug für Matthias, gut genug für mich. Los ging es. Die erste Irritation trat ein, als ich herausfand, dass der GAS keine Register-Aliasse kann. Das bedeutet, ich kann Prozessorregistern keine alternativen Namen geben, um sie darüber anzusprechen. Also z. B. „tos“ für Top-of-Stack. Ich muss den Originalnamen benutzen, z. B. x8. Wieso? Aliasse konnte schon der kleine Shareware-Assembler, den ich in den 80ern für die Entwicklung mit 8051-Mikrocontrollern verwendet habe.

Als GAS mir dann auch noch untersagt hat, 32-Bit-Sprungmarken (Labels) in 16-Bit-Datenfelder zu schreiben (wichtig, um Execution-Tokens platzsparend in Compound-Wörtern unterzubringen), war es aus mit der Liebe. Ich habe mich nach einer Alternative umgesehen.

naken_asm

MICHAEL KALUS hat das *MSP430eForth* mit dem Assembler *naken_asm* seines Namensvetters MICHAEL KOHN übersetzt. Kann der auch RISC-V? Er kann. Allerdings fehlte dem Assembler die sogenannte „Compressed“-Erweiterung des Befehlssatzes für die Unterstützung von 16-Bit-Befehlen. Das ging ja schon gut los. Ich nahm Kontakt mit dem Entwickler auf und fragte, ob er Pläne diesbezüglich hätte. Hatte er nicht, fand die Idee aber interessant und 2 Tage später hatte sein *naken_asm* die Compressed-Opcodes eingebaut. Beeindruckend.

Ich portierte meinen Quelltext rüber zu *naken_asm* und testete ausgiebig. Nach ein wenig Debugging lief der Assembler zufriedenstellend. Aber mir fiel auf, dass *naken_asm* kleine Unzulänglichkeiten hat. Der Assembler unterstützt mehrere Architekturen. Es scheint, als wollte Michael Kohn es allen recht machen und hat ganz unterschiedliche Syntaxregeln eingebaut. Das ist verwirrend. Ich bin da nicht mehr durchgestiegen. Das Aus für den *naken_asm* brachte dann ein Feature des Assemblers, nämlich das Sortieren von Datenblöcken an das Ende der Listing-Datei. Gut gemeint aber völlig unbrauchbar für das Debugging des Dictionarys, bei dem ja Code und Daten gemischt sind. Da kann ich keine Umsortierung gebrauchen. Ich hätte den Maintainer um Support bitten können und vermutlich hätte er sogar das Feature abschaltbar gemacht, aber mir wuchs das alles über den Kopf. Ich wollte keinen Assembler (mit-)entwickeln, sondern ein Forth! Und was, wenn der RISC-V-Teil seines Assemblers doch noch Bugs hatte, die aber erst sichtbar würden, wenn ich in der heißen Phase der Forth-Entwicklung wäre? Hinzu kam, dass ich inzwischen

⁶ Titanic Fans werden wissen, wie das endet.

⁷ Aus dem Englischen: Schimpftirade, Wortschwall

eine Idee hatte, wie ich die Probleme mit dem anderen Assembler (GAS) lösen konnte. Ich wechselte zurück.

Willkommen zurück, GAS

Also portierte ich den aktuellen Entwicklungsstand des Projektes wieder zurück zu GAS. Das hielt sich alles noch in Grenzen. Bis jetzt waren ca. 300 Zeilen Quelltext zusammengekommen.

Die Lösung für meine zwei Probleme mit GAS (keine Aliasse und der Unwille von GAS, Labels in 16-Bit-Daten zu konvertieren) war der C-Präprozessor von GCC. Den kann man nämlich nicht nur auf C-Quelltext loslassen, sondern auch auf Assembler-Quelltext.

Und mit

```
#define tos    x8
#define stack x9
```

habe ich meine Register-Aliasse und kann z. B.

```
sw tos, 0(stack)
```

schreiben, anstatt das erheblich unübersichtlichere

```
sw x8, 0(x9)
```

benutzen zu müssen.

Bei der Lösung des zweiten Problems half mir der Zufall. Ich fand heraus, dass GAS (bzw. der Linker) zwar keine Zuweisung von 32-Bit-Adressen zu 16-Bit Datenfeldern zulässt, wohl aber die Differenz von zwei Adressen (`addr1-addr0`). Wenn ich also dafür Sorge, dass `addr0` Null ist, kann ich das gewünschte Ergebnis „hinmogeln“, indem ich `addr1-addr0` verwende.

Die Sequenz

```
.text # Start of code section
addr0: # This label has to be
        # at address 0x00000000
```

erzeugt ein Label an der Adresse 0.

Mit

```
#define t(x) ((x)-addr0)
```

kann ich dann so etwas machen:

```
# ? ( a-addr -- )
# : ? @ . ;
# Display the contents in a memory cell
Head NOFLAGS, "?"
QUEST: INEST
        .2byte t(AT), t(DOT)
        .2byte t(EXIT)
```

Nicht besonders schön; das MSP430eForth unter `naken_asm` kommt ohne die `t()` Krücke aus.

Im Beispiel ist das Compound-Wort `?` gezeigt. `Head` ist ein Macro und erzeugt den Header des Wortes mit Link- und Name-Feld. `QUEST:` ist ein Label, mit dem das Wort `?` aus anderen Compound-Wörtern mit `.2byte t(QUEST)` aufgerufen werden kann. `INEST` ist ein Macro und ruft

`DOLST` auf. Der Gegenpart ist `EXIT`, das die Liste abschließt. `AT` und `DOT` führen zum Execution-Token von `@` und `.`

CH32V003 — der Chip

Technische Daten

- erhältlich in 4 Gehäusevarianten
 - 8-Pin SOIC
 - 16-Pin SOIC
 - 20-Pin QFN (3x3 mm²)
 - 20-Pin TSSOP
- Speicherausstattung
 - Flash: 16 KB (plus 1920 Bytes Bootloader-Flash)
 - * 2-Byte-Schreibauflösung
 - * 64-Byte-Löschauflösung (Pagegröße: 64 Bytes)
 - * gelöscht Flash hat den Wert \$FFFF
 - RAM: 2 KB
- Taktfrequenz: 48 MHz max.
- Spannungsversorgung: 2,7 V – 5,5 V
- Architektur: RV32EC plus XW-Erweiterung
 - E: abgespeckter Registersatz (16 32-Bit-Register), normal sind 32 Register
 - C: Compressed Extension (16-Bit-Befehlssatz)
 - XW: Herstellerspezifische Erweiterung zur Erhöhung der Codedichte: 8 weitere Load/Store-Befehle mit 16-Bit-Opcodes
- Peripherie (je nach Gehäuseform Einschränkungen)
 - GPIOs: max. 18
 - 1 x SPI
 - 1 x I²C
 - 1 x UART
 - 1 x 10-Bit-ADC
 - 2 x Timer plus 1 x SysTick
 - 2 x Watchdog
 - 1 x Operationsverstärker/Komparator
 - 1 x DMA-Controller
 - 1-Draht-Debug-Interface

Kurioses

Klare Strategie: Der CH32V003 (32-Bit) soll den in China sehr beliebten STM8S003F3 (8-Bit) ersetzen.

- Gleiche Pinbelegung wie der STM8S003F3
- Ähnliche Typenbezeichnung

Entwicklungstools

- Moun River Studio, die offizielle IDE des Herstellers
- WCH LinKE, der vom Hersteller empfohlene Debugger/Programmer
- ch32fun <https://github.com/cnlohr/ch32fun>
 - Komplette Open Source
 - keine Herstellersoftware (Moun River Studio) nötig
 - Auch der Programmer/Debugger vom Hersteller (LinKE) wird unterstützt
 - Viele Beispiele (Startcode, Peripherie) in C
- GNU RISC-V Toolchain
 - Open Source
 - Nicht die tollste Entwicklungsumgebung (optimiert für Compiler-Input)
 - Nach einigen Tricks und mit Präprozessor einigermaßen akzeptabel für den menschlichen Assembler-Programmierer.

Stand des Projektes, Bezugsquelle

RV003eForth läuft und wird von mir bereits in einem Hardware-Projekt eingesetzt und dabei natürlich auch „aus Versehen“ getestet. Ich möchte diese Tests aber noch intensivieren und dafür den sogenannten *ANS Forth tester*, entwickelt von JOHN HAYES, auf eForth anpassen.

Bis zur Veröffentlichung kann es also noch etwas dauern. Aufräumen und Dokumentation stehen an. Wer vorher

schon mal mit dem RV003eForth spielen möchte, kann mir eine E-Mail an wost@ewost.de senden. Oder ihr schaut ab und zu bei <https://codeberg.org/wost> vorbei. Dort wird RV003eForth erhältlich sein.

Pläne

Nun ist also ein neues eForth mit einigem Aufwand entstanden. Was macht man aber nun damit? War es eine Fingerübung, deren einziger Zweck darin besteht, zu zeigen, dass eForth auch auf der RISC-V Architektur kompakt implementiert werden kann?

RV003eForth bittet förmlich darum, in verhältnismäßig kleinen Projekten eingesetzt zu werden, wo Forth aus Kosten- oder Komplexitätsgründen bisher nicht die erste Wahl war. Ich denke da z. B. an digitale Interfaces aller Art. Wie wäre es mit einem Multiprotokoll-Adapter für die verbreiteten 4/8-Bit-LCDs? Das LCD ist ein 5V-Typ? Kein Problem. Der CH32V003 kann auch mit 5V betrieben werden. Kein Spannungs- oder Pegelwandler nötig.

eForth ist ja (auch) als Lehr/Lern-Plattform entwickelt worden. Zusammen mit der günstigen Hardware und moderaten Komplexität des eingesetzten CH32V003 lässt sich ein attraktives Paket schnüren.

Ende

Danke an alle, die sich bis hierhin tapfer durchgelesen haben. Fragen, Anregungen, Ideen, Kritik bitte an wost@ewost.de.

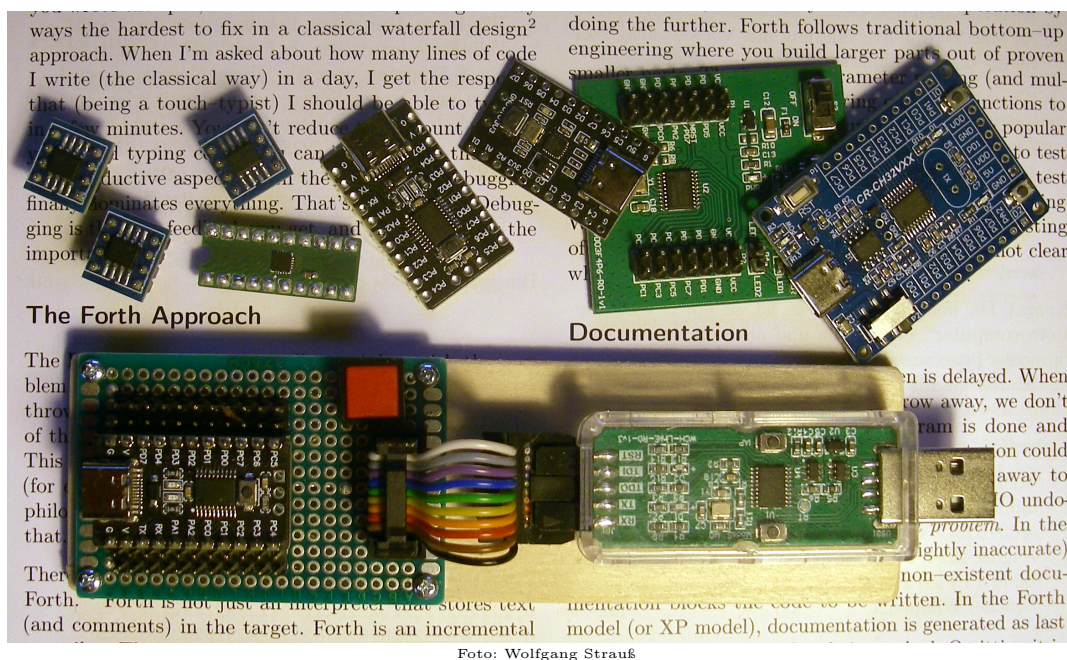


Foto: Wolfgang Strauß

Abbildung 1: Ein Schnappschuss wahllos herumliegender Platinen mit einem CH32V003 drauf. Unten, auf ein Brettchen montiert, meine RV003eForth-Entwicklungshardware. Darauf rechts der WCH Debugger/Programmer „LinKE“, links daneben eines der vielen inzwischen erhältlichen Entwicklungsboards auf einer Trägerplatine

VFD-Modul Noritake

Rafael Deliano

Alphanumerische LCDs sind billig und leidlich gut standardisiert. Es gibt zwar auch größere Typen, aber für groß, hell und guten Ablesewinkel greift man zu VFDs¹.

Die sich deshalb ehemals für Consumer- und Automotive-Anwendungen durchsetzten, typisch aber als kundenspezifische Baugruppen verwendet werden. Ihre japanischen Hersteller Noritake/Itron und Futaba boten immer auch Typen an, die in Ansteuerung und Pinbelegung weitgehend LCDs entsprechen, regulär aber teuer sind. Sie finden sich häufig in Supermärkten an der Kasse als Preisanzeige für Kunden.

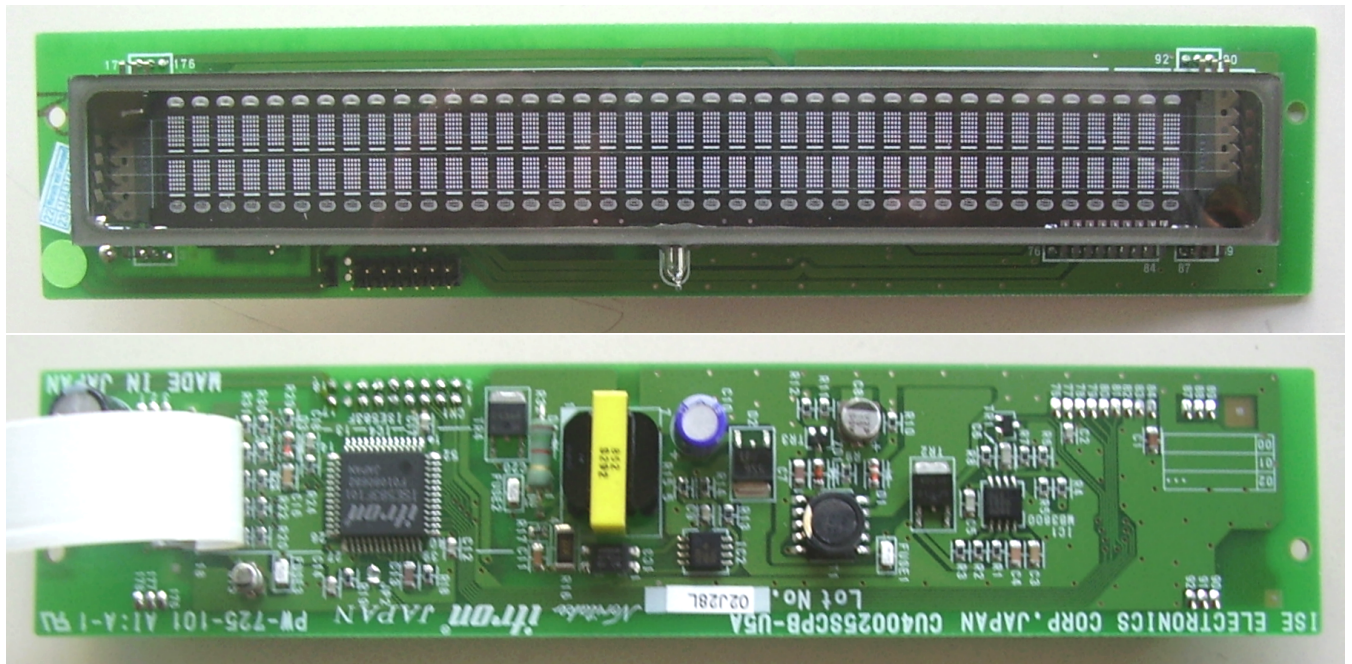


Abbildung 1: Display CU40025SCP-B-U5A a) Aufsicht, b) Unterseite; mittig der Transformator

Das *CU40025SCP-B-U5A* (Abb. 1) war eine teilweise kundenspezifische Variante ohne Datenblatt, die als surplus bei ebay.com deshalb ehemals sehr billig angeboten wurde. Nach etwas Reverse Engineering² fanden sich im www Beschreibungen, wie es zu beschalten ist. Sie tauchen immer noch vereinzelt auf, aber nun für ca. 50 EUR.

Die zwei Zeilen zu 40 Zeichen belegen 16x2 cm². Die äußeren Abmessungen sind etwas größer, 21x5,5 cm². Allerdings ist die Tiefe mit 33 mm durch den Trafo des DCDC üppig.

Kabel

Vorgesehen auf der Unterseite ist ein 18-pol. Folienkabel mit 1mm-Pitch — für Prototypen mühsam. Alternativ sind aber im Layout seitlich auch Bohrungen vorhanden, mit praktisch gleicher Pinbelegung, für 2-reihige Stiftleisten. Die wurden nachbestückt und dann mit einer 20-pol. IDC-Buchse³ und Flachbandkabel verwendet (Abb. 2).

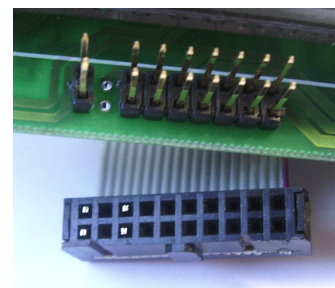


Abbildung 2: Flachbandkabel

Schaltung

Der Stecker hat die von den LCDs gewohnte Pinbelegung zur Datenübertragung (Abb. 3) und zwei abgesetzte Pins für die Stromversorgung der Heizspannung des LCDs. Diese ist galvanisch getrennt (Abb. 2) und mag ursprünglich für 12 Vac ausgelegt gewesen sein.

¹ Vacuum Fluorescent Display

² Um die Komponenten zu identifizieren.

³ Insulation Displacement Connector (auf Deutsch: Isolationsverdrängender Kontakt). Auch „Schneidklemmen“ genannt.

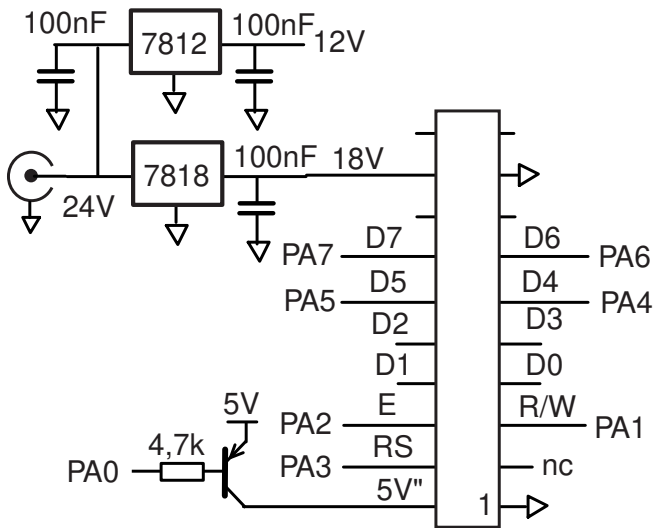


Abbildung 3: Schaltung Breadboard

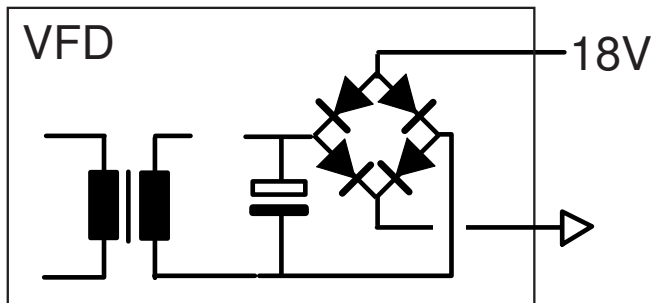


Abbildung 4: Versorgung 18 V

Die Heizspannung am VFD ist 5 Vac (Abb. 5) und an den Verbindungen zum Glas messbar.

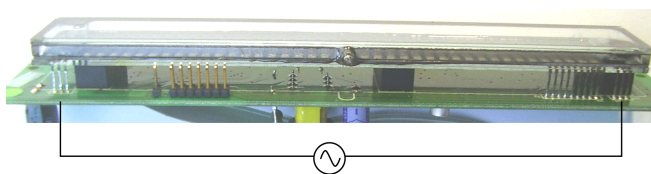


Abbildung 5: Messung der Spannung am Heizer

Im Breadboard war es günstiger, aus einem 24Vdc-Steckernetzteil 18Vdc zu erzeugen. Die Stromaufnahme ist mit 150...170 mA etwas abhängig davon, wie viele Buchstaben leuchten. Der Spannungsanstieg beim Powerup sollte steil sein, sonst startet der DCDC nicht — moderne Labornetzteile sind übrigens oft zu langsam!

Da die 24 V für den 5V-Regler des AW60 zu hoch sind, wurde hier noch ein 7812 vorgesehen, der sie auf 12 V reduziert. Der Controller auf der Display-Platine bleibt bei fehlerhafter Ansteuerung oft stehen. Er hat keinen Resetpin auf der Schnittstelle. Deshalb sind in meiner Schaltung seine 5 V schaltbar. Die Stromaufnahme ist gering, etwa 4 mA.

Die Ansteuerung in Software ist kompatibel zum HD44780 Controller der LCDs. Nur die Startup-Sequenz ist in Details spezifisch (s. Listing). Genau wie bei LCDs kann man auch 16 Sonderzeichen im RAM definieren.

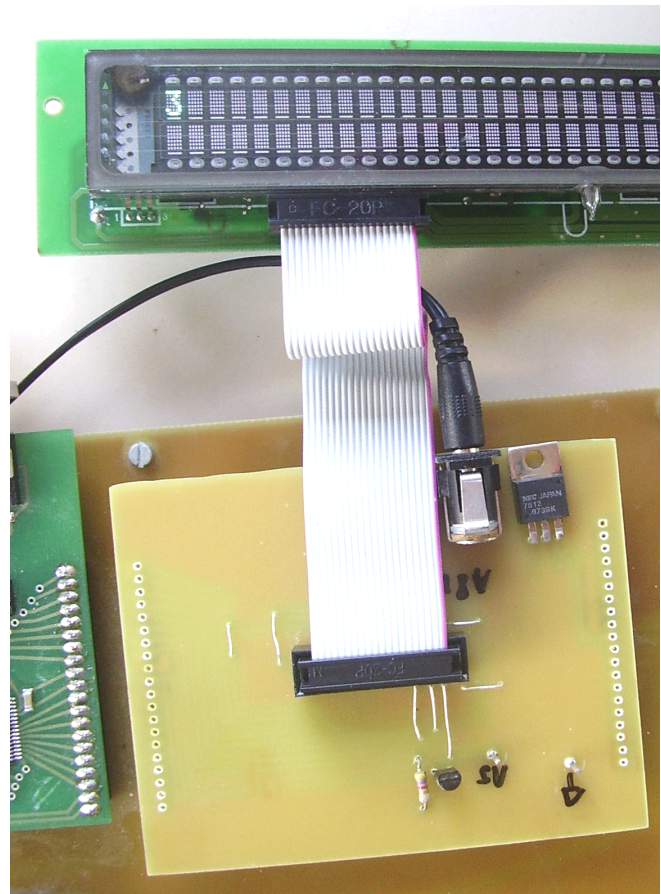


Abbildung 6: Breadboard am AW60

Test

INIT-LCD CLEAR startet die Anzeige. Dann kann man ASCII-Zeichen direkt in den Anzeigen-Speicher schreiben (Tab. 1), die damit auch angezeigt werden:

33 0 LCD! \ "3" to 0

Zeile	Bereich (hex)
obere	00 ... 27
untere	40 ... 67

Tabelle 1: Lage des Anzeigen-Speichers

Anwendung

Ideal für alle Geräte, die bei hellem Tageslicht verwendet werden. Bei Batteriebetrieb sind große LCDs jedoch sinnvoller. Mit einem Retro-AIM65/40 als Steuergerät ist das ein guter Blickfang auf jeder Maker-Faire — aus etwas Entfernung besser lesbar als jeder Laptop!

Links

- <https://www.noritake-elec.com/products/vfd-display-module>
- https://www.youtube.com/watch?v=c-RD_Xw0i5U

Listing

```

1  <| \ SRCO
2
3  PA 0 DCONSTANT /EN-5V      \ o 1
4  \ PA 1 DCONSTANT LCD_R/W   \ o 0
5  PA 2 DCONSTANT LCD_EN     \ o 0 CS
6  PA 3 DCONSTANT LCD_RS     \ o 0 Adr
7  PA 4 DCONSTANT LCD_DO_D4   \ o 0
8  PA 5 DCONSTANT LCD_D1_D5   \ o 0
9  PA 6 DCONSTANT LCD_D2_D6   \ o 0
10 PA 7 DCONSTANT LCD_D3_D7   \ o 0
11
12 : INIT          \ ( --- )
13   01 PA C! FF DPA C! D% 10 MSEC 00 PA C! ;
14
15 \ -----
16
17 : PULSE-EN     LCD_EN B1! LCD_EN B0! 1 MSEC ;
18
19 : i4LCD!       \ ( bbbb0000 --- ) RS=0
20               PA C! PULSE-EN ;
21
22 : iLCD!        \ ( UC1 --- ) RS=0 instruction
23   DUP FO AND   PA C! PULSE-EN
24     4<SHIFT    PA C! PULSE-EN ;
25
26 : dLCD!       \ ( UC1 --- ) RS=1 data
27   DUP FO AND  B% 1000 OR PA C! PULSE-EN
28     4<SHIFT  B% 1000 OR PA C! PULSE-EN ;
29
30 : LCD!        \ ( UC1 addr --- )
31               \ addr: 00 ... 0F ; 40 ... 4F
32   80 OR iLCD! dLCD! ;
33
34 : CLEAR      B% 0001 iLCD! 2 MSEC ;
35               \ LCD off, Curson off , Blink off
36 : LCD-OFF    B% 1000 iLCD! ;
37               \ LCD on, Curson off , Blink off
38 : LCD-ON    B% 1100 iLCD! ;
39
40 : BLANK      \ ( --- )
41   OF 00 DO 20 I LCD! LOOP
42   4F 40 DO 20 I LCD! LOOP ;
43
44 : INIT-DISPLAY \ ( --- )
45   B% 00110011 i4LCD! 5 MSEC
46   B% 00110010 i4LCD! 5 MSEC
47   B% 00000000 i4LCD! \ 08 Display on/off
48   B% 10000000 i4LCD!
49   \ DCB D=1 Display off,
50   \ C=0 Cursor off
51   \ B=0 Blink off
52   B% 00000000 i4LCD! \ 01 Clear Display
53   B% 00010000 i4LCD!
54   B% 00000000 i4LCD! \ 06 Entry Mode Set
55   B% 01100000 i4LCD!
56   \ I/D S
57   \ I/D=1 increment cursor
58   \ S=0 no shift Display
59
60   \ 0C LCD on, Curson off , Blink off
61   B% 00000000 i4LCD!
62   B% 1100 i4LCD!
63   B% 00000000 i4LCD!
64   B% 0010 i4LCD! ;
65
66 : INIT-LCD    \ ( --- )
67   INIT
68   INIT-DISPLAY LCD-OFF BLANK BLANK LCD-ON ;
69
70 : LCD-FILL    \ ( UC1 --- )
71   27 00 DO DUP I LCD! LOOP
72   67 40 DO DUP I LCD! LOOP DROP ;
73
74 CR ." -> SRC1.TXT"
75 CR |>
76

```

Zur Geschichte von Noritake

„Die Noritake Itron Corp. (Ise Electronics Corp.), eine Tochtergesellschaft der japanischen Noritake Co., Ltd., erfand 1967 die Vakuumfluoreszenzanzeige (VFD). Diese Anzeigetechnik wurde der Weltöffentlichkeit durch einen Artikel in der Zeitschrift „Electronics“ vom 29. Mai 1967 vorgestellt. Die erste entwickelte einstellige Anzeigeröhre trug in den 1970er-Jahren zur Entwicklung elektronischer Taschenrechner bei. Nach der Entwicklung der VFD mit flacher Glasstruktur 1974 wurden ihre Anwendungsbereiche auf Schreibmaschinen, Haushaltsgeräte, Audio- und Videogeräte, Fahrzeugpanels, Industrieanlagen, Spielautomaten, Medizintechnik und Kommunikationsgeräte ausgeweitet.

Anfangen hat die Firma im späten 19. Jahrhundert. Damals öffnete sich die bis dahin abgeschottete japanische Gesellschaft dem internationalen

Handel. Baron ICHIZAEMON MORIMURA, ein bedeutender Vertreter des japanischen Handels jener Zeit, erkannte das Exportpotenzial in die Vereinigten Staaten. 1876 gründete er die Firma Morimurakumi, um *Porzellan* und andere Geschenkartikel nach Amerika zu exportieren. Um die höchste Qualität seiner Exporte zu gewährleisten, beschloss er, die Produktion selbst zu kontrollieren und eine eigene Fabrik zu errichten. Zu diesem Zweck gründete er am 1. Januar 1904 in *Noritake*, einem Dorf nahe Nagoya, die Firma Nippon Toki Gomei Kaisha ...“ [Basierend auf der ins Deutsche übersetzten Darstellung der Firmengeschichte von Noritake (Quelle: <https://www.noritake-elec.com/about/history>)]

Interessant fand ich auch die chronologische Tabelle von Noritake Electronics, in der sie sehr schön die technische Entwicklung der *VFD tubes* 1967 bis hin zum großflächigen *glass capacitive touch VFD module GU-D Series* und darüber hinaus aufzeichnen. mk

Forth-Gruppen regional

Bitte erkundigt euch vorab bei den Veranstaltern, ob die Treffen stattfinden.

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-Neustheim

München **Bernd Paysan**
Tel.: (0 173) – 82 06 874
bernd@net2o.de
Treffen: Jeden 4. Donnerstag im Monat um 20:00 auf <http://public.senfcall.de/forth-muenchen>, Passwort **over+swap**, im Sommer auch wieder um 19:00 im La Capannina, Weiltstraße 142, 80995 München

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
Derzeit keine Treffen.

Dienste der Forth-Gesellschaft

Nextcloud <https://cloud.forth-ev.de>

GitHub <https://github.com/forth-ev>

Twitch <https://www.twitch.tv/4ther>

µP-Controller-Verleih **Carsten Strotmann**
microcontrollerverleih@forth-ev.de
mcv@forth-ev.de

Spezielle Fachgebiete

Forth-Hardware in VHDL **Klaus Schleisiek**
microcore (uCore) Tel.: (0 58 46) – 98 04 00 8_p
kschleisiek@freenet.de

KI, Object Oriented Forth, **Ulrich Hoffmann**
Sicherheitskritische Systeme Tel.: (0 41 03) – 80 48 41
uho@forth-ev.de

Forth-Vertrieb **Ingenieurbüro**
volksFORTH **Klaus Kohl-Schöpe**
ultraFORTH Tel.: (0 82 66) – 36 09 862_p
RTX / FG / Super8
KK-FORTH

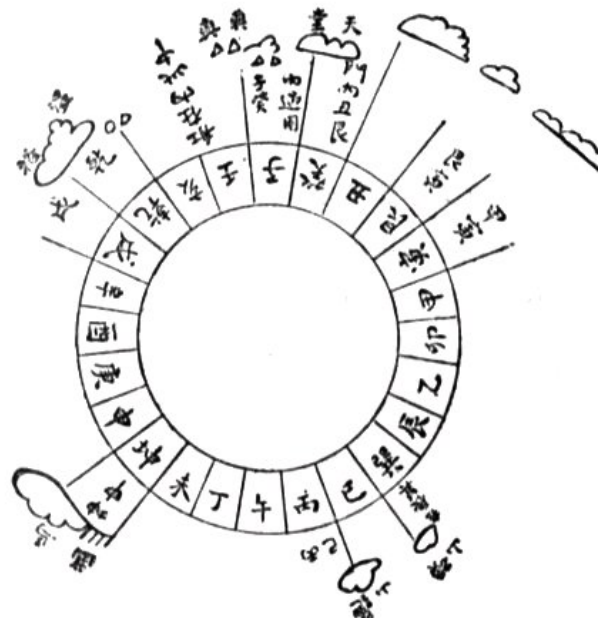
Termine

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*

Jeder 1. Montag im Monat ab 20:30 Uhr
Forth-Abend
Videotreffen (nicht nur) für Forthanfänger
Info und Teilnahmelink: E-Mail an wost@ewost.de

Jeder 2. Samstag in ungeraden Monaten
ZOOM-Treffen der Forth2020 Facebook-Gruppe
Infos zur Teilnahme: www.forth2020.org

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.

Bald isses so weit ...¹

Michael Kalus

Bei der Forth-Tagung neulich in Burladingen zeichnete es sich ab, die nächste Tagung wird im schönen Bochum (NRW) sein. „Tief im Westen, wo die Sonne verstaubt ...“, dichtete GRÖNEMEIER einst auf Bochum, just als wir anfangen mit dem Forth-Magazin „Vierte Dimension“. So ist das wohl, Jubiläen ab dem Vierzigsten² feiert man in Bochum. 2026 wird es dann schon unser 42. Jahrgang sein.

Unsere Tagungsstätte 2026: Bochum

So hab ich mich also umgesehen und bin in der Mitte von Bochum auf die DJH Jugendherberge gestoßen, die sich über die Jahre zu einer respektablen Tagungsstätte gemacht hat (Abb. 1). Hab mir das also mal selbst angesehen. Die Tagungsräume sind tatsächlich modern und mit bester Technik, so wie wir es brauchen — Beamer, schnelles Internet, alles da. Der Service ist gut. Frühstück und Mittagessen im Haus; Obst, Kaffee, Kuchen in den Pausen. Die Zimmer sind schlicht, aber bequem. Und es hat auch genug Einzelzimmer für diejenigen unter euch, welche das suchen. Aber auch Mehrbettzimmer, die es preiswerter machen. Ich denke, das geht, oder?



Foto: Raenmaen

Abbildung 1: Die DJH Jugendherberge Bochum

Am Wochenende nach Ostern 09. – 12. April 2026

Anvisiert ist nun, traditionell, *das Wochenende nach Ostern*. Ich hoffe, das sagt den meisten von euch zu und wir haben eine gut besuchte Tagung.

¹ ... würde man in Bochum sagen :)

² Das letzte große Grönemeyer-Konzert in Bochum fand am 12., 13., 15. und 17. Juni 2024 im Vonovia Ruhrstadion statt. Diese Konzerte waren Teil seiner Tour zum 40-jährigen Jubiläum des Albums „4630 Bochum“ aus 1984.

³ „27plus“ — älter als 27 Jahre.

⁴ Das Bermuda3Eck ist das wohl bekannteste Kneipenviertel im gesamten Ruhrgebiet und zieht Menschen aus der ganzen Region an. Es bietet eine vielfältige Auswahl an Lokalen für jeden Geschmack.

Kosten

Die Preise für uns³ liegen zwischen 54,50 € im 4-Bett-Zimmer und 105,00 € für eine Einzelperson im 2-Bett-Zimmer, jeweils pro Nacht mit Frühstück.

Und die Tagungspauschale pro Person wird um die 50 € sein, worin neben dem Raum und dessen Ausstattung auch schon das Mittagessen, die Erfrischungsgetränke und Obst, Kaffee und Kuchen für die Pausen enthalten sind. Abendessen haben wir auch im Haus. Und Bars die 24 Stunden geöffnet sind: Das Bermuda-Dreieck⁴ ist nebenan.

Vorbestellung

Du kannst bei mir schon jetzt dein Zimmer vorbestellen. Schreib mir einfach eine E-Mail an mik.kalus@gmail.com.

In Kürze wird auch die Online-Anmeldung eröffnet auf www.forth-ev.de

Wir sehen uns!

Anreise

Die DJH Jugendherberge Bochum ist etwa 2 km vom Bochumer Hauptbahnhof entfernt. Mit dem Taxi sind das ungefähr 6 Minuten.

Parkhäuser für den eigenen Wagen sind da überall und rund um die Uhr geöffnet.

Und wer will, kann über Dortmund einfliegen, von da sind es etwa 35 km zum Tagungsort, mit dem Auto ungefähr 30 Minuten und es gibt eine direkte S-Bahn-Verbindung nach Bochum, die Fahrt dauert etwa 40 Minuten.

Auch über den Flughafen Düsseldorf ginge das, die Fahrt mit der Bahn von dort zum Bochumer Hbf dauert etwa 50 Minuten.