Vierte Dimension

4/2022

Das Forth–Magazin

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

In dieser Ausgabe:

Titelbild Tzolkin–Abschnitt im Codex Dresdensis, beginnend mit dem Tag 1 Manik. Gezeichnet von LACAMBALAM. (Ausschnitt bearbeitet für das Heft: mk)

`www.flickr.com/photos/lacambalam`

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: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Liebe Leser,

Wilhelm Busch dichtete neulich: „Eins, zwei, drei! Im Sauseschritt — Läuft die Zeit; wir laufen mit. . . . Max und Moritz, diese Knaben, sollen, hör' ich, Eltern haben. Einen der und eine die, — Nämlich Scherz und Phantasie. . . .“

Mögen euch solche Eltern erhalten bleiben in diesen Zeiten. Und natürlich der elektrische Strom. Ohne den kein Forth–Magazin. Jedenfalls nicht in dieser Form.

Wie man Zeit zählt, hat uns Menschen schon seit langem beschäftigt. Folgt man Bernd Paysan bis zu den Links, kommt man auch zum Tzolkin–Kalender, wovon ein kleiner Ausschnitt zum Titelbild geworden ist. Zuvor sehen wir aber noch, wie ihr es anstellen könnt, dem Endbenutzer in einem fernen Land euer Forthprogramm schmackhaft zu machen.

Bis man dazu in der Lage ist, braucht es einen gewissen Anlauf. Forth im Embedded–Bereich zu vermitteln, hat sich Wolfgang Strauss auf die Fahne geschrieben und Mitstreiter gewonnen. Seht selbst, wie weit das *Feuerstein–Projekt* inzwischen gediehen ist. Mitmachen kann man natürlich auch.

Im Fall des Falles wollt ihr das sogar auf selbstgebautem Einplatinencomputer bewerkstelligen? Dann ist Rafael Deliano euer Mann! Dazu braucht es heute nicht mehr viele Bauteile. Und wie man die eigene Hardware dann forthig testet, hat er auch gleich angegeben.

Das komplette Forth durchtesten, vom Kern bis zu den High–Level–Worten, vor dieser Aufgabe steht derjenige immer mal wieder, der ein eigenes Forth neu aufsetzen muss. Sei es, dass eine neuere MCU benutzt werden soll, oder ein Wechsel auf einen besser verfügbaren Chip erzwungen wird, weil Lieferketten versagt haben. Jörg Völker schildert den Vorgang am Beispiel seines *FancyForth*.

Wer weiß noch, dass in diesen kleinen handlichen *WikiReadern* auch ein Forth werkelte? Klaus Kohl–Schöpe hat damit seinen Urlaub versüßt und Schach im hellsten Sonnenschein auf das Display gezaubert, zumindest die Züge seiner Schach–Literatur, die mitgereist war. Ein Display auch für Feuersteine? Das Faltblatt in der Heftmitte ist übrigens auch von ihm. Und ihr bekommt die alle auch bei ihm!

Die *Mitgliederversammlung der Forth–Gesellschaft* wird im Jahre 2023 als Videokonferenz vor Ostern stattfinden. Ich hoffe, ihr seid alle dabei — reisen muss man dafür nicht mehr. Diese Frühlings–Videokonferenz entwickelte sich inzwischen zur eigentlichen Forthtagung. Details dazu gibt es traditionell auf der „Rückseite“ im Heft. Im Sommer dann soll es zusätzlich auch ein leibhaftiges Forthtreffen geben. Haltet Ausschau danach.

Auf ein weiteres forthiges Jahr mit vielen inspirierenden Forth–Treffen!

Und nun erstmal *Fröhliche Weihnachten* euch.

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-2022-04

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

Ulrich Hoffmann Kontakt: Direktorium@Forth-ev.de
Bernd Paysan
Gerald Wodni

Mecrisp–Quintus 1.0.0 jetzt stabil!

Es ist soweit! Nach vier Jahren Entwicklungszeit und 38 experimentellen Zwischenständen hat *Mecrisp–Quintus*, ein optimierendes Forth für RISC–V und MIPS, nun mit der heute[1] veröffentlichten *Version 1.0.0* den Meilenstein *Stabilität* erreicht.

Es hat eine Weile gedauert, doch nicht jedes Forth kommt mit Unterstützung für

- zwei unterschiedliche Architekturen,
- eigenen FPGA–Designs,
- hausgemachten Prozessoren,
- astronomischen Berechnungen und
- Fouriertransformation

daher. Der Registerallokator *Acrobatics*, bei Mecrisp–Stellaris RA ein undurchdringliches Knäuel Spaghetti–Code in Assembler, ist diesmal lesbar und nachladbar in Forth selbst geschrieben und hält — sofern möglich — nicht nur die Elemente des Datenstacks, sondern auch jene des Returnstacks in Registern.

Außerdem liegen mehrere *Emulatoren* bei, nicht nur zum Erstellen von Kernen mit vorkompilierten Forth–Quelltexten (was für ein alter Hut) — wer mag, kann sogar einen RISC–V–Mikrocontroller seinen eigenen Prozessor emulieren lassen und so Definitionen in Maschinensprache in Einzelschritten und mit Registereinblick ausführen, wofür der Kontext für den Benutzer transparent zwischen dem echten und dem emulierten Prozessor ausgetauscht wird.

Mamihlapinatapai[2] schließlich erlaubt es ganz großen Fans von RISC–V, Mecrisp–Quintus sogar auf ARM–Mikrocontrollern auszuführen, wobei allerdings Mecrisp–Quintus und Mecrisp–Stellaris schon von Haus aus weitestgehend kompatibel sind.

Vielen Dank an all die Menschen, die Ideen beigetragen, Wünsche geäußert und Fehler gemeldet haben!

Matthias Koch

2023: Wie alt wird Forth eigentlich?

> „FORTH, Inc. has been providing innovative, reliable, custom software development services and firmware engineering services under contract for more than forty years.“ (`https://www.forth.com/`)

Genauer gesagt seit 1973 — 49 Jahre ist das nun her. Zunächst auf Minicomputer ausgerichtet hielt *FORTH* bald auch Einzug bei den *Mikros*. So ging das los:

EDN 1976

In der Ausgabe vom 20. Nov. 1976 der US–Fachzeitschrift EDN[3] ging es nur um ein Thema: *Mikroprozessoren*. Es war das 3. Jahr, in dem das „Microprocessor Directory“, eine Aufstellung aller 40 verfügbaren Typen, publiziert wurde. Davon sicher 20% Vaporware, Prozessoren, die nie lieferfähig wurden. Nach den Großfirmen drängten nun auch kleinere Anbieter wie MOS Technology auf den Markt. Der Fokus bewegte sich im „Microcomputer Systems Reference Issue“ langsam weg von den Chips hin zu Systemen und Software. In dem Bereich tummelten sich die Kleinunternehmen. Wie die Journalisten im Heft launig vermerkten: „Nothing beats the low overhead of a cottage corporation“. Für die ist ein klappriges Logo und marktschreierischer Superlativ im Text kennzeichnend. Forth Inc. existierte zwar schon seit 1973, war aber auf Minicomputer ausgerichtet und hatte bestenfalls im Bereich Astronomie eine gewisse Bekanntheit. Das scheint auf die Raumfahrt abgefärbt zu haben.

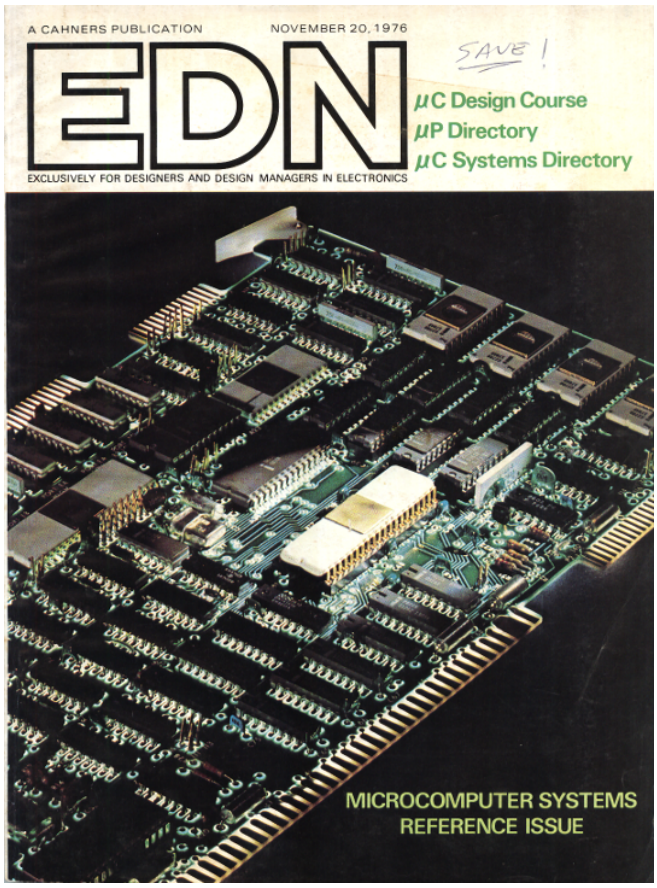


Abbildung 1: EDN, 20. November 1976 — Micro Computer Reference Issue

[1] 19. Okt. 2022

[2] Das schöne Wort „Mamihlapinatapai“ kenne ich aus dem Buch „Prinzessin Insomnia & der alptraumfarbene Nachtmahr“ von Walter Moers, nach dessen Lektüre auch Mecrisp–Quintus selbst einst zum Namen kam. Wikipedia schließlich verriet mir, dass dieses Wort in der Sprache der Ureinwohner Feuerlands tatsächlich existiert, mit der ganz besonderen Bedeutung „das Austauschen eines Blickes zwischen zwei Personen, von denen jeder wünschte, der andere würde etwas initiieren, was beide begehren, aber keiner bereit ist, zu tun“ — Und ist es nicht genau das, wenn RISC–V auf ARM läuft? (Matthias)

[3] Electronic Design News (EDN) is an electronics community for engineers, by engineers. `https://www.edn.com/`

RCA

Der Pionier bei Schallplatten und Fernsehen hatte als Großunternehmen seine besten Jahre schon hinter sich.[4] Der Bereich Halbleiter war jedoch noch erfolgreich und brachte 1969 die CMOS–Logik CD4000 auf den Markt. Weil alteingesessen, finanzierte RCA seine R&D[5] teilweise aus Geld von Pentagon und NASA. Die wollten einen CMOS–Mikroprozessor, vorzugsweise rad–hard. Die Technik war noch nicht so weit, die Chipgröße üppig, der Takt niedrig. RCA baute erstmals 1975 den Chipsatz CDP1801R und CDP1801U. Anfang 1976 folgte der Einchipprozessor CDP1802. Es war zwar der erste CMOS–Mikroprozessor der Welt[6], man kam aber recht spät auf den Markt und war chronisch leistungsschwach. RCA hatte für Consumer–Elektronik einen guten Namen und brachte bald auch Einplatinencomputer mit tinyBASIC heraus. Von der RCA Semiconductor Division kam 1976 der Auftrag an *Forth Inc.* für den neuen Chip ein FORTH zu implementieren [1], wohl nicht für den Hobby–Kunden gedacht. Es wurde letztlich anscheinend nicht von RCA vertrieben [2]. Forth Inc. portierte ihr microFORTH auf Intel 8080, Motorola 6800 and Zilog Z80 und war damit recht erfolgreich. Ein früher Spin–off war Karl Meinzers IPS auf dem CDP1802 für AMSAT[7] [3]:

> „A language that meets most of the points just mentioned has been published by CH Moore under the name of FORTH. Particularly, Moor's implementation technique is very ingenious. On the other hand, his user interaction mechanisms were too limited for our problem area. Thus, I rather took FORTH as a platform, from which I designed and implemented my own system which I named IPS."

Es ging später auch in den Fundus von figFORTH über und erreichte damit die privaten Anwender [4]:

> „Judging by the numerous Forth articles that began to appear in the pages of Ipso Facto, figForth was a hit with 1802 computer enthusiasts."

Rafael Deliano

Quellen

[1] Bergin, Gibson „History of Programming Languages" Addison–Wesley 1996
[2] „RCA may offer memory–savings processor language" Electronics Feb. 19 1976
[3] Meinzer „IPS An Unorthodox High Level Language" BYTE Jan. 1979
[4] `www.cosmacelf.com`

„Listings are hard to follow"

Im Kapitel Mikroprozessoren von [5] findet sich eine Aufstellung der damals relevanten Programmiersprachen. Neben Basic, Fortran, Cobol, C wurden als „other languages" auch ALGOL, APL, Pascal, FORTH, LISP, MUMPS, PILOT, PL/1, RPG–II aufgeführt, mit kurzen Bemerkungen zu den Sprachen.[8]

> „Forth is an extendable, stack–oriented language, developed for data acquisition and radiotelescope orientation at an astronomical observatory. Its program listings are hard to follow.[9] Users may define their own instructions in FORTH."

Rafael Deliano

Quelle

[1] Fink Electronic Engineers Handbook 3.ed McGraw-Hill 1989

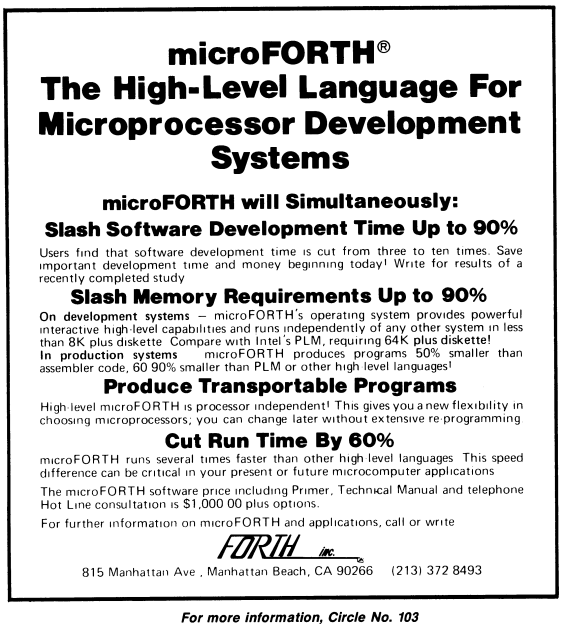


Abbildung 2: Erste microForth–Anzeige

[4] Der Begriff RCA steht für: Radio Corporation of Amercia. Sie brachten 1940 Audiosteckverbinder heraus, die unter dem Namen *Cinch* bekannt geworden sind. Das englische Wort „Cinch" bedeutet so viel wie „Fest im Griff".

[5] research and development — Forschung und Entwicklung.

[6] `http://www.antiquetech.com/?page_id=682`

[7] AMSAT ist eine Vereinigung von Funkamateuren, die das Ziel verfolgen, Raumfahrtsatelliten zu betreiben. Die 1969 gegründete Organisation hat ihren deutschen Sitz in Bochum.

[8] Und so ist es auch heute noch. Forth läuft immer noch unter „Sonstige". :)

[9] Da es so frei formbar ist, liegt auch die Gestaltung des Quellcodes ganz in der Verantwortung des jeweiligen Schöpfers. Wir hoffen, durch das Forth–Magazin immer wieder Anregungen gegeben zu haben, wie das lesbar sein kann.

Das Paper von Moore und Leach

Abbildung 3: Ein neuer Stern im Fokus von NRAO: FORTH

Nimmt man die Rechnung der *EuroForth 2018*, dann sind's 2023 schon 55 Jahre Forth. Aber ich rechne eher mit *1970,* oder noch besser *1971* als Startpunkt.

1970, weil da das Paper von Moore und Leach herauskam [1]. Ich hatte mir neulich dazu notiert:

> „Describes Forth, as it was in 1970. There are surprising differences from and surprising similarities with modern Forth systems. The system they describe uses text interpretation instead of threaded code for definitions, although there is already a code field, i.e., the foundation for indirect threading. During the interpretation of a definition, only words defined earlier are visible, like in modern Forths, and in contrast to Postscript. To make text interpretation speed bearable, the dictionary is implemented as a hash table with external chaining. There is support for portably generating CODE words (called verbs in the paper). The syntax is a bit more complicated than today: special characters may only come as first character in a word, so words are not only separated by spaces. The system already features multitasking (round–robin, with preemption). Source is stored in screens (then called sheets) containing 50 lines by 40 characters. The programs look markedly different than today because the primary stack manipulation words are `@T` (similar to `PICK`) and `=T` (similar to a word sometimes called `STICK`). Forth was running on the IBM–1130 and the Burroughs B–5500 (different cell sizes). The paper also observes that compactness of programs 'arises through the economies of tailoring definitions to a specific application', and is more pronounced in larger programs."

Angesichts dessen, dass jenes Forth noch mit `@T` und `=T` arbeitete, sehe ich es noch als Vorstufe zum eigentlichen Ur–Forth, das aber wohl kurz danach entstanden ist, denn das Kitt–Peak–Forth hatte schon `DUP SWAP DROP OVER` , die Glen Haydon als Teil von Level 0 von Forth identifiziert hat [2][3]. Und Kitt–Peak–Forth ist meines Wissens ein Ergebnis der Arbeit von Moore für die Astronomie (ab 1971 am NRAO[10]). Also ist 1971 das bessere Jahr als 1970?

Weitergehende Papers zu dem Thema sind „Forth – The Early Years" von Chuck Moore [4], wo die einzelnen Komponenten von Forth mit Jahreszahlen genannt werden.

Und zwei Papers mit dem Namen „The Evolution of Forth": Eines von Moore in Byte (August 1980) [5], wo Moore selbst schreibt:

> „When I invented FORTH about 10 years ago . . . "

und das erste moderne Forth *nach* seiner Arbeit auf der IBM 1130 bei Mohasco ansetzt. Er fährt fort:

> „The first use of modern FORTH occurred when it was written for a Honeywell H316 at the NRAO . . . "

Also 1971. Da steht auch, dass in dieser Stufe der *Return Stack* und das *Dictionary* als Datenstruktur dazugekommen sind.[11]

Im anderen Paper von Rather, Colburn, und Moore, in „History of Programming Languages II", 1993 publiziert [6], steht es auch so geschrieben:

> „Moore developed the first complete, stand–alone implementation of Forth in 1971 for the 11–meter radio telescope operated by the National Radio Astronomy Observatory (NRAO) at Kitt Peak, Arizona."

Anton Ertl

Quellen

[1] Charles H. Moore and Geoffrey C. Leach, „FORTH — A Language for Interactive Computing"; Mohasco Industries, Inc.; 1970
http://www.ultratechnology.com/4th_1970.html
[2] http://kittpeak.forthfiles.net/Kitt-Peak-Forth-Primer-1979.pdf
[3] http://www.forth.org/literature/forthlev.html
[4] http://worrydream.com/refs/Moore%20-%20Forth%20-%20The%20Early%20Years.pdf
[5] http://www.inventio.co.uk/The_Evolution_of_FORTH,_an_Unusual_Language_2018Jan18.pdf
[6] https://www.forth.com/resources/forth-programming-language/#1_Chuck_Moore8217s_Programming_Language

[10] Das National Radio Astronomy Observatory ist eine staatlich geförderte Forschungs– und Entwicklungsorganisation für Radioastronomie mit Sitz an der University of Virginia in den USA.

[11] Wenn ich es recht verstehe, waren die Namen vorher noch hartcodiert im Programmcode, aber so genau ist das nicht beschrieben.

Einst und jetzt

Abbildung 4: Aktuelle Werbeanzeige in Make:

Die Heise *Make:* hat sich gut bei den Lesern etabliert. Aber die Zeiten haben sich geändert: weniger Abos, weniger Ausgaben/Jahr, weniger Inserenten als anno Elrad. Also wenig Kleinanzeigen. Und die werden deshalb nun vom Verlag für den Kampfpreis von 150 EUR beworben, weil viel leerer Platz zu füllen ist (Abb. 4). Die Zahl ist geschönt, bezieht sich auf den Jahrespreis 6x150 = 900 EUR. Einzelanzeige wäre 260 EUR. Die wird aber leicht übersehen, das Schicksal einer Kleinanzeige eben. Deshalb ist in der Werbung die teure Schrotflinte üblich.

Das Format ist inhaltlich restriktiv: *Logo* + 300 Zeichen Text. Da Anzeigen prägnant sein sollen, vielleicht kein so großer Nachteil. Mehr als eine passende Catchphrase[12] zu Forth und „kostenloses Probeheft anfordern“ wäre nicht nötig.

Man kann sich dafür aufhübschen: aus geeigneten alten Artikeln ein „Maker–Sonderheft“ verfertigen und zusammen mit einer aktuellen VD an Interessenten verschicken.

Kostet alles Geld. Zu der Einsicht, dass konsequente Öffentlichkeitsarbeit nicht kostenlos ist, hat sich der Verein nie durchringen können.

Die Idee, eine Anzeige in auflagenstarker Zeitschrift zu schalten, um potentielle Mitglieder zu erreichen, gab es vor langer Zeit schon mal. Anno Elrad wäre eine Anzeige schwarz–weiß statt farbig und deutlich teurer gewesen. Eine Idee damals war ein Tausch. D. h., eine kleine Anzeige der VD in Elrad gegen vier große Anzeigen der Elrad in der VD — wurde nie weiterverfolgt. Die Auflage beider Zeitschriften unterschied sich auch damals krass. Heise wäre absehbar nicht darauf eingegangen.

Unklar ist, wie groß die Schnittmenge der Leser von Make: und der Mitglieder des Vereins ist. Klar ist aber, dass alle Aktivitäten für Hobby–Mikrocontroller jetzt unter *Maker* einsortiert werden.

Rafael Deliano

Nicht–Forthiges, aber nahe dran!

Ich bin über einen Artikel gestolpert, in dem der Autor Jay Carlson beschreibt, wie er mit N parallelen Entwicklungsprojekten umgeht. Der hat sich einen Arbeitsplatz eingerichtet, da wirste blass um die Nase. Sehr ausgefuchst. Vielleicht ist das eine Inspirationsmeldung wert[13] [1]:

Mir schwebt tatsächlich vor, eine Mini–Ausgabe davon zu erstellen. Mit vielleicht 4 Tabletts aus Sperrholz, die mit ESD Matte bezogen werden. Da werden dann so Sachen wie USB–Hub drauf befestigt (angeschraubt) und die Controller–Platinen ebenfalls. Vielleicht wird das mit dem Bastelkram dann stabiler.[14]

Und — wie macht ihr das so?

Erich Wälde

[1]`https://jaycarlson.net/2021/09/18/juggle-embedded-projects-home-office-workspace-tour/`

Forth–Quelltext für die Fourier–Ansicht eines Signals

Weil nachgefragt wurde: Den haben wir schon abgedruckt im vorigen Heft (4d2022–03), siehe die Definition `fft-plot` in `asciisignal.txt`, ab Zeilennummer 350. Und Matthias Koch ergänzte in unserer Korrespondez darüber:

> „Falls du auf die Implementierung der Fouriertransformation selbst neugierig bist, die liegt natürlich Mecrisp–Ice bei, im Verzeichnis `common/Fourier`. Der Algorithmus hat eine trickreiche Skalierung, damit das mit maximaler Genauigkeit auf 16–Bit–Ganzzahlen läuft! Und was die FFT an sich angeht, gibt es hier eine schöne Internetseite: `https://www.katjaas.nl/FFT/FFT.html` “

Lieber Matthias, vielen Dank für den Link. Hat mir die Mathematik hinter der FFT einen großen Schritt näher gebracht. Vielleicht schaffe ich es ja doch noch eines Tages, Laie der ich bin, die FFT komplett zu verstehen.

M. Kalus

Jostein Skjelstad †

Uns erreichte kürzlich eine traurige Nachricht:

> „Jostein Skjelstad, wohnhaft in Sand, Norwegen, ist leider am 28.9.2022 verstorben.
>
> Zwei Artikel/Programme wurden von Jostein im Forth–Magazin gedruckt. Im Heft 4d2017–01 ‚Vintage Computing — FORPS‘ und dann im Heft 4d2018–03 ‚Integer und lineare Gleichungen‘.

[12] Schlagworte

[13] Aber ja!

[14] Bin selbst auch auf dem Weg, umzugestalten. Ein feines Ambiente hat sich Jay Carlson da geschaffen. Mikro–Elektronik mit Stil. Hut ab. (mk)

Ich half Jostein bei der Übersetzung ins Deutsche. Etwa 10 Tage, bevor er plötzlich, aber nicht unerwartet an Krebs starb, bat er mich, sein neustes Program ins Deutsche zu übersetzen, damit es vielleicht im Forth–Magazin erscheinen könne.

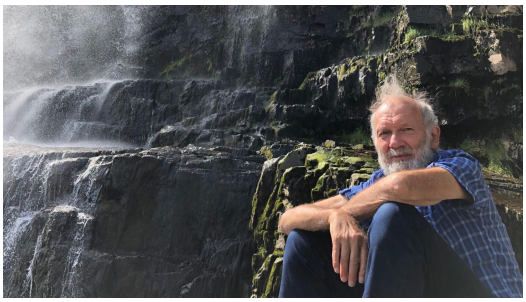


Abbildung 5: Jostein Skjelstad

Jostein arbeitete viele Jahre als Landvermesser in unserer Gemeinde — aber nebenbei vertiefte er sich in die Entwicklung von Programmen, die unter anderem beim Landvermessen nützlich sein können. Die Programme, die er entwickelte, halfen ihm und anderen bei der Arbeit.

An seinem letzten Programm hat er längere Zeit gearbeitet und sah sich gezwungen, zwei verschiedene Werkzeuge zu benutzen, weil kein Programm alles konnte, was er brauchte. So kombinierte er Tcl/Tk und Forth. Tk ist eine grafische Benutzeroberfläche zu Tcl. Das Programm zeigt die „von Hand“ gemessenen Daten als einen grafischen Plot.

Die Beisetzung war am 11.10.2022.

Mit freundlichen Grüßen, Michael Willig“

Herr Willig sandte den Entwurf des Artikels und Josteins Code zu. Dem Wunsch, bzw. letzten Willen zur Publikation, kommen wir insofern nach, als das Werk nun im Forth–Wiki steht. Die Erklärungen zum Code sind in Deutsch, der Code selbst ist in Norwegisch verfasst. Ich konnte das daher nicht selbst sichten, verstehe kein Norwegisch. Doch vielleicht sind einige Leser dazu in der Lage und finden Anregungen darin. mk

Link: Projekt „Hanakam“.
`https://wiki.forth-ev.de/doku.php/various:various`

Beste leden van HCC!Forth en in Forth geïnteresseerde computeraars

Leon Konings, Vorsitzender des HCC!Forth, schrieb Anfang Dezember in der Einladungs–E–Mail zum Treffen:

„Aufgrund der Aufhebung fast aller Corona–Maßnahmen gibt es jetzt keine Einschränkungen außer der Kapazität des Raumes (ca. 22 Personen).“[15]

Inzwischen hat das physische Treffen am Samstag, den 10. Dezember 2022, von 11:00 bis 15:00 Uhr stattgefunden. Doch auch nachträglich möchte ich mitteilen, dass diese Forthgruppe in den Niederlanden munter werkelt. Interessierte sind jederzeit willkommen. Die Treffen sind immer in der *Boslaan 1a* neben der *Zuiderkapel Bilthoven*. mk

`https://forth.hcc.nl/agenda`

Abbildung 6: Hübsche Gegend da an der Booslaan ...

MovForth

nannte Xuyang Chen das Release–V0.0.2–alfa vor einem Jahr. Es sei ein „LLVM frontend for the Forth Language“. In seinem Readme war zu erfahren:

„MovForth compiles Forth source code to executable binaries. Using LLVM IR as an intermediate target, it is an experiment in adapting Forth for modern compiler libraries and modern architectures.

- Bottom–up compilation; no dictionary or interpreter remains in final executable
- Compile time evaluation for immediate words allows programmers to use meta–compilation without fear
- Compiling to LLVM IR allows for compilation to pretty much any architecture
- Modern LLVM optimization passes used on Forth code

You can find Forth source and its corresponding compiled forms in Examples/... “

Bin gespannt, ob jemand damit Erfahrungen gesammelt hat? cas/mk

`https://github.com/Reschivon`

[15] Original: „Door de opheffing van vrijwel alle corona–maatregelen zijn er nu geen beperkingen meer behalve de capaciteit van de zaal (ongeveer 22 personen).“

Internationalization mit Gforth (und Forth–200x)

Bernd Paysan

Stephen Pelc *und* Peter Knaggs *haben vor Jahrzehnten einen Vorschlag zur Internationalisierung von Forth–Programmen (für Endbenutzer) präsentiert [1], aus dem Teilkonzepte auch als RfD*[1] *in den Forth200x–Prozess eingeflossen sind. Die Diskussion um diesen hier behandelten Teil ist leider eingeschlafen, aber Gforth implementiert die wesentlichen Teile des Vorschlags; das wird in MINΩΣ2 auch tatsächlich genutzt. Was das ist und wie man es benutzt, erklärt dieser Artikel.*

Internationalization (18) und Localization (10)

Ein Programm für Endbenutzer richtet sich an Leute, die es in ihrer Muttersprache bedienen wollen. Insbesondere GUI–Programme sollten das können. Es geht dabei im Wesentlichen um Texte für Bedienfelder und Meldungen. Üblicherweise werden die Texte aus der Entwicklersprache (meist Englisch) nicht vom Entwickler selbst übersetzt, sondern vom Vertriebspartner im jeweiligen Land, oder, bei freier Software, von Nutzern aus dem Land, die zwar Englisch können, aber keine Experten in Programmierung oder gar Forth sind.

Aus diesem Ablauf ergibt sich die Notwendigkeit, dass das Programm selbst die Übersetzung nicht einfach enthält; das Programm wird beim Übersetzen nicht angefasst, sondern nur die Liste der Texte. Dafür gibt es auch in anderen Programmiersprachen und GUI–Frameworks Vorbilder, etwa `GNU gettext` [2].

Andere Teile des Vorschlags von 1999 sind inzwischen (ggf. auch deutlich anders) im Standard angekommen, etwa das Xchar–Wordset (das die Zeichensatzfrage löst) oder Substitute (das Vorlagen ausfüllt, bei denen z. B. die Reihenfolge von der Sprache abhängt).

Konzepte

Locale

Ein Locale repräsentiert eine Sprache und mit Sprachvarianten verknüpfte Details, wie das Land, in dem die Sprache gesprochen wird. So ist der erste Monat des Jahres im Deutschen (de) „Januar“, in Österreich (de_AT) aber „Jänner“. Die aktuelle Jahreszahl[2] z. B. ist in weiten Teilen der Welt 2022, in Taiwan (zh_TW) aber 111, und in Israel (iw_IL) 5783.

Der Vorschlag enthält drei Komponenten, die zusammen ein Locale bilden: Sprache, Land und Encoding. Gforth implementiert nur zwei dieser Komponenten, da das Encoding generell UTF–8 ist.

Wie in der Juristerei gilt bei Locales, dass der Spezialfall den allgemeinen Fall überschattet. Eine landesspezifische Eigenart überdeckt die allgemeinere Regel.

LSID (Locale String IDentifier)

Da es sich hier um Texte dreht, die übersetzt werden müssen, ist das gebräuchliche Objekt ein Locale String IDentifier, kurz LSID. Dieser String ist in der Entwicklersprache (zumeist Englisch) im Quelltext des Programms abgelegt, und von dort aus kann man auf die Übersetzungen zugreifen. Ein LSID ist ein opaker Datentyp, in Gforth ein Index. LSIDs werden mit `L" <string>"` erzeugt, wobei gleiche Strings die gleiche LSID zur Folge haben; man also den gleichen Text an mehreren Stellen im Listing einfach mit der gleichen Stringkonstante im `L"` referenzieren kann.

Die Übersetzung einer LSID erhält man mit `LOCALE@ ( lsid -- addr u )`, den String in der Entwicklersprache mit `NATIVE@ ( lsid -- addr u )`.

Grundsätzliche Implementierung

Gforth implementiert Locales als Array of Strings, und einer Verkettung zum nächstgenerischen Locale (also von Sprache+Land zu Sprache zu generischem Locale zu Entwicklersprache). Eine LSID ist der Index in das Array. Die LSID 0 ist damit zulässig; damit es nicht zu Konfusion kommt, weil 0 oft als spezieller Wert angesehen wird, ist die LSID 0 vom System schon reserviert (für den Namen des Systems selbst).

Wird die gesuchte LSID im aktuellen Locale nicht gefunden, wird also zum nächsten darüberliegenden Locale gegangen, bis zur Entwicklersprache.

Übersetzungen einlesen

Der Vorschlag in Forth200x enthält keine standardisierte Möglichkeit, Übersetzungen einzubringen; das war „außerhalb des Standards“. So ist das Ganze natürlich wenig nützlich; es ist nicht möglich, lokalisierte Standard–Programme zu schreiben. Gforth fügt dem ein `LOCALE! ( addr u lsid -- )` hinzu, mit dem man einzelnen LSIDs im aktuellen Locale einen String zuweisen kann, als Primitive für eigene Interfaces. So könnte man z. B. eine Tabelle als CSV anlegen, und jede Sprache hätte eine eigene Spalte.

[1] Request for Discussion

[2] Wobei man für die Datumskonversion ohnehin verschiedene Algorithmen braucht (auch der Neujahrspunkt und die Länge der Monate unterscheiden sich), und die Vorlage dann die passenden Felder enthält.

Für ganze Locales gibt es `INCLUDE-LOCALE ( "name" -- )`, mit dem man eine Datei laden kann, die für jede LSID eine Zeile enthält — die Zeilennummer ist dabei die LSID. Das ist zwar fragil (ändert man den Quelltext und fügt neue LSIDs ein, muss man diese Dateien auch nachführen), aber sehr einfach zu implementieren. Alternativ könnte man auch eine Syntax ähnlich wie bei GNU gettexts `.po`–Dateien einführen, bei der der unübersetzte Text als Index für den übersetzten Text dient. Hier muss man aber ebenfalls nacharbeiten, wenn sich der unübersetzte Text ändert. Das Verwenden einer Tabelle ist vielleicht für das Warten von Programmen einfacher — insbesondere, weil die CSV–Tabelle selbst dann mit einem Tabellenprogramm gepflegt werden kann, etwas, was viele Leute beherrschen.

Beispiel: Datumsausgabe

Oben geschildertes Datum–Problem möchte ich als kleines Beispiel nutzen. Wir brauchen zunächst ein paar Libraries.

```
require i18n.fs
require substitute.fs
```

Damit können wir die Zeit entsprechend umwandeln, und dann ausgeben.

```
: >%time%&%date%&%tz% ( udns -- )
  #1000000000 um/mod swap >time&date&tz
  s" tz" replaces  2drop
  dup 0 <# # # # # #> s" YYYY" replaces
  dup 0 <# #s #> s" year" replaces
  \ for Taiwan
  #1911 - 0 <# #s #> s" twyear" replaces
  dup 0 <# # # #> s" MM" replaces
  0 <# #s #> s" month" replaces
  dup 0 <# # # #> s" DD" replaces
  0 <# #s #> s" day" replaces
  dup dup #12 >=
  IF  #12 - s" pm"  ELSE  s" am"  THEN
  s" a/pm" replaces
  dup 0= IF  #12 +  THEN
  0 <# #s #> s" 12h" replaces
  0 <# # # #> s" hh" replaces
  0 <# # # #> s" mm" replaces
  0 <# # # #> s" ss" replaces
  l" %YYYY%-%MM%-%DD%T" locale@
  $substitute drop s" date" replaces
  l" %hh%:%mm%:%ss%" locale@
  $substitute drop s" time" replaces ;

: .date&time ( dtime -- )
  >%time%&%date%&%tz%
  l" It is %time% %tz% on %date%" locale@
  .substitute drop ;
```

Das funktioniert natürlich schon, nur ist es nicht lokalisiert.

```
cr ntime .date&time
It is 13:37:00 CET on 2022-11-19T ok
```

Also lokalisieren wir das mal direkt vom Forth aus, und legen dazu erst mal Sprachen und Länder an:

```
language en
en country en_UK country en_US country en_IN
language de
language zh
zh country zh_TW
```

Jetzt können wir uns an das Lokalisieren machen. Dazu nutzen wir, dass der L"–String ja immer die gleiche LSID ergibt.

```
l" It is %time% %tz% on %date%" >r
de s" Es ist %time% %tz% am %date%" r@ locale!
zh s" 现在是%date%%time%%tz%" r@ locale!
zh_TW s" 現在是%date%%time%%tz%" r> locale!
```

Die Formate der Datumsausgabe unterscheiden sich natürlich auch je nach Sprache und Land (was besonders bei Englisch verwirrend ist, weil sich zwischen US– und UK–Englisch die Reihenfolge von Monat und Tag ändert).

```
l" %YYYY%-%MM%-%DD%T" >r
en_US s" %MM%/%DD%/%YYYY%" r@ locale!
en_UK s" %DD%/%MM%/%YYYY%" r@ locale!
en_IN s" %DD%-%MM%-%YYYY%" r@ locale!
de s" %day%.%month%.%year%" r@ locale!
zh s" %year%年%month%月%day%日" r@ locale!
zh_TW s" %twyear%年%month%月%day%日" r> locale!
```

Auch die Uhrzeit hat lokale Varianten, wobei wir hier nur die englische mit am/pm implementieren wollen.

```
en s" %12h%.%mm%.%ss%%a/pm%" l" %hh%:%mm%:%ss%"
locale!
```

Ausprobieren!

```
default-locale to locale ok
: .dt 2dup .date&time ; ok
ntime ok
.dt It is 15:49:40 CET on 2022-11-20T ok
de .dt Es ist 15:49:40 CET am 20.11.2022 ok
en .dt It is 3.49.40pm CET on 2022-11-20T ok
en_US .dt It is 3.49.40pm CET on 11/20/2022 ok
en_UK .dt It is 3.49.40pm CET on 20/11/2022 ok
en_IN .dt It is 3.49.40pm CET on 20-11-2022 ok
zh .dt 现在是2022年11月20日15:49:40CET ok
zh_TW .dt 現在是111年11月20日15:49:40CET ok
2drop ok
```

Was fehlt noch?

Eine lokalisierte Zahlenausgabe, natürlich. Das lässt sich zum Teil mit Strings machen (für arabische und indische Schriften mit einem positionsabhängigen Dezimalsystem [3] könnte man mit einer Übersetzung von `L" 0123456789"` arbeiten), für Chinesisch braucht man einen Algorithmus, weil die Stellenwerte dort mitgeschrieben werden, und Stellen mit Wert 0 weggelassen werden (ein positionsabhängiges Zahlensystem, eine Art Abacus auf Papier, wurde 2000 Jahre lang, bis vor ca. 500 Jahren,

parallel benutzt, hat sich aber nie allgemein durchgesetzt).

Datumsangaben sind oft teilnumerisch, und verwenden Monatsnamen, die dann eine Liste von 12 lokalisierten Namen benutzen sollten.

Dann gibt's noch alternative Kalender, etwa den chinesischen Mondkalender, den arabischen und hebräischen Kalender, sogar neue, wie den indischen Nationalkalender, und noch einige andere [4]. Das sind alles Dinge, die man nicht mit einfachen Textübersetzungen machen kann, dafür braucht man jeweils angepasste Algorithmen. Das zeigt dann die Grenzen des hier vorgestellten Verfahrens auf.

Referenzen

[1] Stephen Pelc, Peter Knaggs, *RfD: Internationalisation,* `http://forth200x.org/internationalisation.txt`

[2] GNU Gettext Project, *Gettext Manual,* `https://www.gnu.org/software/gettext/manual/gettext.html`

[3] Wikipedia, *Liste anderer Ziffern,* `https://en.wikipedia.org/wiki/Arabic_numerals#Comparison_with_other_digits`

[4] `https://de.wikipedia.org/wiki/Liste_der_Kalendersysteme`

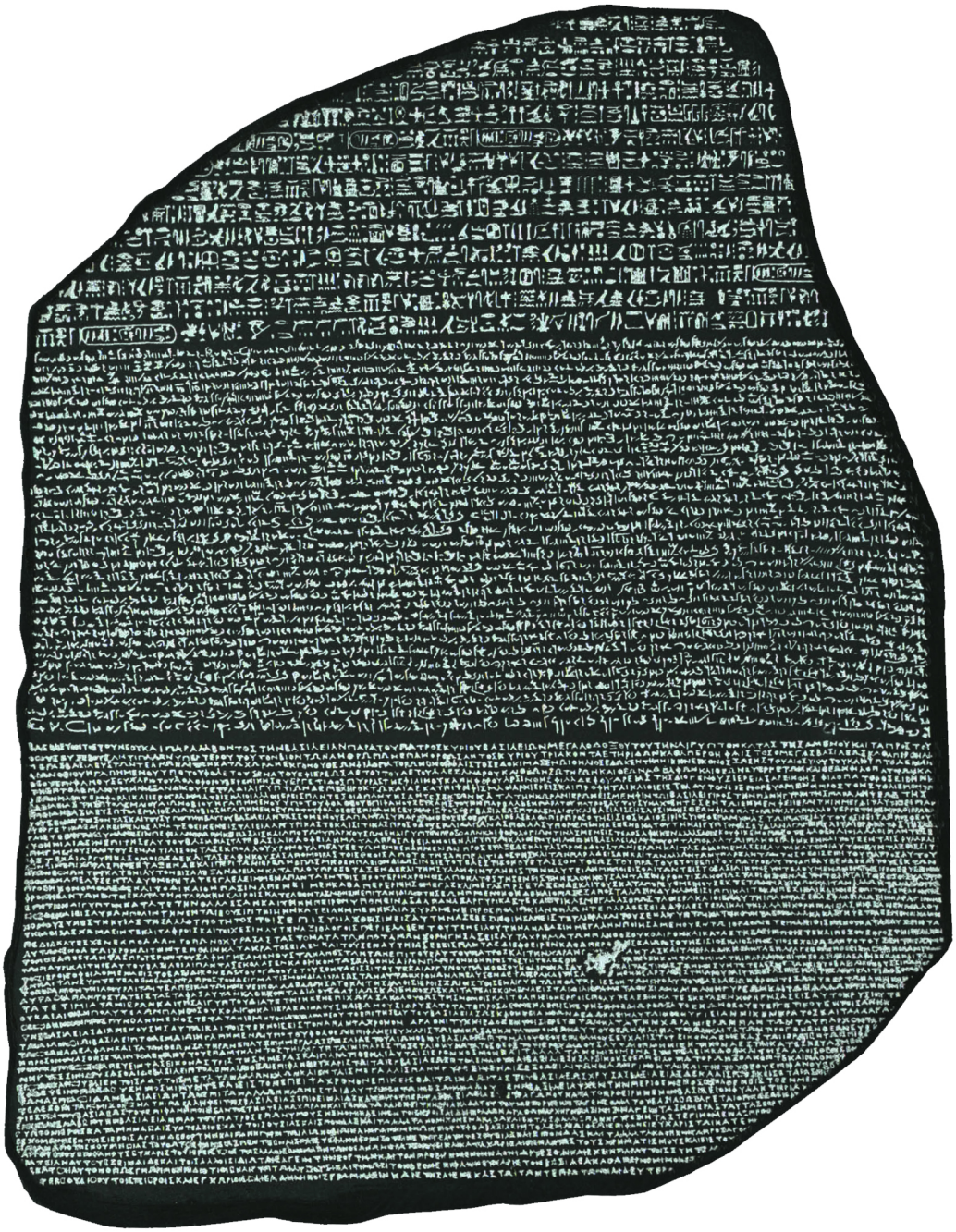


Abbildung 1: Einer der ersten Nachweise von i18n und l10n

Projekt Feuerstein

Wolfgang Strauß

Auf der virtuellen Forth–Tagung 2020 habe ich die Idee für ein neues Projekt vorgestellt und um Mitarbeit geworben. Das Projekt mit dem Namen „Feuerstein“ soll es Forth–Einsteigern erleichtern, mit der neuen Materie vertraut zu werden. Der Fokus liegt auf der Arbeit mit Mikrocontrollern. Typische Hindernisse und Fallstricke sollen vermieden werden. Die Benutzererfahrung steht ganz oben auf der Liste. Das zu erstellende Paket hat den Anspruch, komplett zu sein (Hardware, Software, Dokumentation).

Kurz nach der Tagung hat ein kleines Team die Arbeit aufgenommen. In den folgenden 1,5 Jahren fanden über 80 Videotreffen zum Thema statt, in denen diverse Ansätze besprochen, Ideen ausprobiert und der Rahmen des Projektes festgelegt wurde. Was nun zu tun ist, ist eine Fleißarbeit. Die Menge an gesammelten Informationen muss aufbereitet und in eine Form gebracht werden, die der Öffentlichkeit vorgestellt werden kann.

Im Folgenden möchte ich im Frage/Antwort–Stil einige Aspekte des Projektes beleuchten.

Wieso Forth?

Die Wahl von Forth als Programmiersprache für Mikrocontroller scheint obligatorisch zu sein für ein Mitglied der Forth–Gesellschaft. Tatsächlich wird die Frage aber regelmäßig von Interessenten auf z. B. Messen gestellt. Wieso nicht einfach MicroPython nehmen, da ist alles so schön einfach? Kann man machen. Mir aber gefällt an Forth die Transparenz bis hinunter zum Maschinencode und der Minimalismus der Implementierung eines kompletten Systems in wenigen Kilobytes. Die Möglichkeit der unbegrenzten Erweiterung des Systems nach exakt meinen Vorstellungen ist ein Bonus, den ich nicht mehr missen möchte. Dafür nehme ich den Aufwand der eigenen Erstellung von Peripherietreibern in Kauf.

Wieso der Name „Feuerstein“?

Ein Projekt muss einen „knackigen“ Namen haben. Feuerstein gefiel mir auf Anhieb. Er symbolisiert den Funken, der überspringt. Mit dem Begriff „Stein“ kann man auch schön spielen. Der Baustein, den man programmiert oder eine Serie von Boards mit Namen von Mineralien etc.

Wieso RISC–V?

Das Projekt hat einen Mikrocontroller mit RISC–V ISA[1] als Protagonisten. Ich wollte den Schwung mitnehmen, den RISC–V gerade hat. RISC–V hat Potential und eine große Zukunft. Feuerstein auch? You bet!

Wieso der Baustein GD32VF103?

Der GD32VF103 der Firma GigaDevice war der erste praxistaugliche Mikrocontroller mit RISC–V–Kern auf dem Markt. Er ist im Prinzip eine peripherieseitig aufgeblasene Version des massig verwendeten und gut bekannten Veteranen STM32F103, nur halt mit dem „coolen“ Kern. Die englische Dokumentation ist dem chinesischen Hersteller bemerkenswert gut gelungen und es lässt sich vernünftig damit arbeiten.

Welche Boards werden unterstützt?

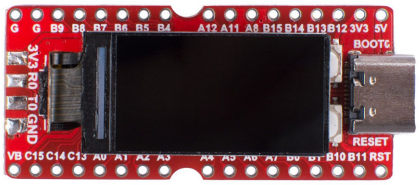


Abbildung 1: Longan Nano

Feuerstein wird primär auf dem Longan Nano (Abb. 1) entwickelt. Das Board ist preiswert und (auch während der aktuellen Chipknappheit) gut erhältlich. Ein kleines Farb–Display und ein mikroSD–Kartenhalter laden zum Experimentieren ein.

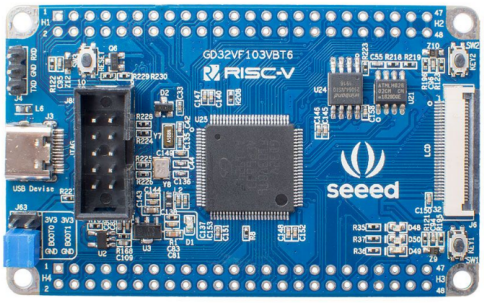


Abbildung 2: SeeedStudio GD32–Dev–Board

Ein weiterer supporteter Baustein ist das Seeed Entwicklungsboard (Abb. 2). Es hat üppig IO, ein I²C–EEPROM und 8 MByte serielles Flash. Außerdem einen Anschluss für ein LCD zur Entwicklung von Benutzerschnittstellen. Aufgrund des externen Speichers können Anwendungen realisiert werden, welche die 128 KB des Controllers überfordern würden.

[1] Instruction Set Architecture. Gemeint ist der Befehlssatz mit Namen, Funktionen und Kodierung.

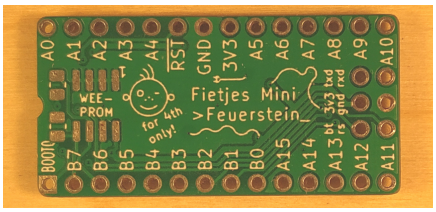


Abbildung 3: Fietje Mini

Das dritte derzeit unterstützte Board (Abb. 3) stammt von Ingolf Pohl. Er hat es sich nicht nehmen lassen, eine eigene Platine mit dem Formfaktor einer DIL28–Fassung für das Feuerstein–Projekt zu entwickeln. Nachdem ich scherzhaft bemerkte, dass jetzt nur noch ein I²C–EEPROM fehlt, hat er das auch noch im ohnehin engen Design untergebracht. Respekt! Es ist der optional zu bestückende Footprint im Bild. Die restlichen Bauteile befinden sich auf der Unterseite der Platine, damit oben genug Platz für die Pinbeschriftung ist.

Wieso eine eigene Platine?

Wenn man einen Standard–Baustein haben will, der auch Jahre später noch unverändert verfügbar sein soll, bleibt wohl nichts anderes übrig, als es selbst zu machen. Auf die großen Hersteller ist kein Verlass. Als Beispiel soll unser geliebter Longan Nano der Firma Sipeed dienen.

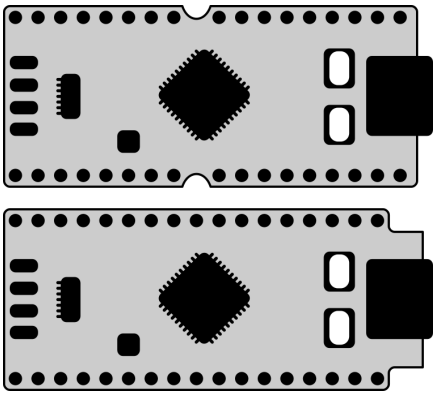


Abbildung 4: Zwei Versionen der Longan–Nano–Platine

Schaut euch mal Abb. 4 an. Oben die erste Version, unten der „verbesserte“ Nachfolger. Die Schaltung ist dieselbe, aber der Hersteller hat es sich nicht nehmen lassen, den Umriss der Platine zu ändern. Im Rahmen der Umgestaltung ist dann auch die rechte Hälfte der Pins um ein Rasterloch nach links gewandert, um die Lücke zu schließen. Die Belegung ist aber ansonsten gleich geblieben. Geht's noch? Alle Erweiterungsplatinen, die den „alten“ Longan Nano als Modul aufgenommen haben, müssen nun umgebaut werden oder man darf mit einem Adapter arbeiten. Alle Dokumentationen müssen nun zwei Varianten auseinanderhalten. Danke, Sipeed!

Wieso Mecrisp–Quintus?

Tja, für das Feuerstein–Projekt nur vom Feinsten. Will sagen, das Forth von Matthias Koch hat eine Qualität, die ihresgleichen sucht. Ich kenne nichts Besseres. Außerdem ist der Support von Matthias unübertroffen gut.

Wieso VIS?

Auch hier gilt das Prinzip: „Das Beste ist gerade gut genug.“ VIS ist eine Erweiterung für Mecrisp–Quintus (und einige andere Forth–Versionen) von Manfred Mahlow, die an die bekannten Forth–Vokabulare erinnert, aber anders funktioniert und weit über deren Möglichkeiten hinausgeht. Mit VIS ist es u. a. möglich, elegante Datenstrukturen zu bauen.

Wieso ist die Dokumentation in Englisch?

Der erste Gedanke beim Schreiben einer Dokumentation ist wohl, sich die Arbeit nicht unnötig schwer zu machen und deshalb einfach die Muttersprache zu verwenden. Die Welt aber spricht Englisch. Eine eventuelle internationale Zusammenarbeit ist nur auf Englisch möglich. Warum dann nicht beides? Englisch und Deutsch? Das würde bedeuten, zwei Versionen der Dokumentation aktuell zu halten. Das wäre eine (zu?) große Aufgabe für so eine kleine Gruppe.

Wer ist Zielgruppe für Feuerstein?

Feuerstein hat primär den neugierigen Forth–Interessierten im Sinn. Jemanden, der einen schnellen und unkomplizierten Einstieg zum ersten Ausprobieren sucht, um herauszufinden, ob dieses eigenartige „Forth“ etwas für ihn ist. Feuerstein hat aber nicht den Anspruch, dem Benutzer jegliche Arbeit abzunehmen. Nach dem Schnelleinstieg, bei dem es eher um die spielerische Beschäftigung mit dem System geht, wartet auf den Benutzer jede Menge Lernstoff und die Heranführung ans eigenständige Arbeiten mit Forth. Durch die fixierte Umgebung (Hardware, Software, Dokumentation) kann bei Problemen leichter Hilfe geleistet werden.

Warum dauert das so lange?

Gut Ding will Weile haben. Der Anspruch von Feuerstein ist der leichte Einstieg in Forth auf Mikrocontrollern. Das bedeutet, die erste veröffentlichte Version muss funktionieren. Es gibt da keine zweite Chance.

Der zweite Grund ist, dass ich selbst auch noch lerne. Es gibt eine Menge Werkzeuge rund um das Projekt, in die ich mich einarbeiten muss.

- Versionsverwaltung: Git, GitHub, SourceHut
- Dokumentationssystem: Sphinx
- Platinenlayout: KiCAD
- Forth: Mecrisp–Quintus, VIS
- Wikipedia–Seiten: Forth–e.V.–Wiki
- Assembler: RISC–V
- Grafik: Inkscape, Gimp, Krita

Wieso nur für Mikrocontroller?

Das Projekt Feuerstein hat inzwischen einen respektablen Umfang angenommen. Da ist die freiwillige Reduzierung der behandelten Themen sinnvoll, um das Projekt handhabbar zu halten.

Doch keine Sorge: Für Forth auf dem PC gibt es ein eigenes Vereinsprojekt.

Was ist in der Firmware des Chips?

- Spezialversion von Mecrisp–Quintus
- Assembler
- Disassembler
- VIS
- History–Buffer
- Hilfesystem

Entwicklungssystem auf dem PC?

- Terminal e4thcom von Manfred Mahlow
- FEE (Forth Enhanced Editor von Ingolf Pohl)
- Firmware Uploader

FEE?

FEE (Forth Enhanced Editor) ist eine übersichtliche Umgebung von Ingolf Pohl für die Forth–Entwicklung auf Mikrocontrollern. Das herausstehende Merkmal ist, wie Forth–Quelltext zum Mikrocontroller geleitet wird. Um genauer zu sein: welche Teile des gerade im Editor geöffneten Quelltextes zum Controller gelangen. Dazu werden in den Text sogenannte „Stopper“ eingefügt. Bei einer Übertragung sendet FEE den Text des durch Stopper definierten Textblocks rund um den aktuellen Cursor über die serielle Schnittstelle zum Chip. Damit bekommt das interaktive Arbeiten mit Forth eine zusätzliche Dimension.

Welche Betriebssysteme werden unterstützt?

Die Entwicklung findet auf GNU/Linux statt, und zwar auf PC und Raspberry Pi. Ob Microsoft Windows offiziell unterstützt wird, steht noch nicht fest. Ich habe zwar noch einen Windows–Rechner, aber Spaß macht der nicht. Am Anfang des Projektes dachte ich, die Unterstützung von Windows wäre wichtig, um die Reichweite zu erhöhen. Inzwischen bin ich mir da nicht mehr so sicher. Wir werden sehen ...

Was ist an Dokumentation geplant?

- Schnelleinstieg (erste Schritte) für den Ungeduldigen
- Buchreihe Mecrisp–Quintus: Tutorial, Benutzerhandbuch, Referenzhandbuch
- Buch: Der Mikrocontroller GD32VF103
- Buch: Kochbuch/Rezepte
- Hardware–Benutzerhandbuch der unterstützten Boards
- Datenblätter/Benutzerhandbücher der verwendeten Bausteine

Wo kann man schauen?

Im Wikipedia–Bereich auf der Website der Forth–Gesellschaft gibt es einen Artikel mit Informationen rund um das Projekt Feuerstein.

`https://forth-ev.de/wiki/projects:feuerstein:start`

Dort findet ihr auch Kontaktmöglichkeiten zum Team und Links zu Dokumenten und weiteren Quellen, die mit dem Projekt zu tun haben.

Ausblick

Oh ja, es ist noch viel zu tun. Das ist mir beim Schreiben dieses Artikels wieder bewusst geworden. Macht aber nichts. Nur immer aufpassen, dass der Spaß nicht zu kurz kommt. :-)

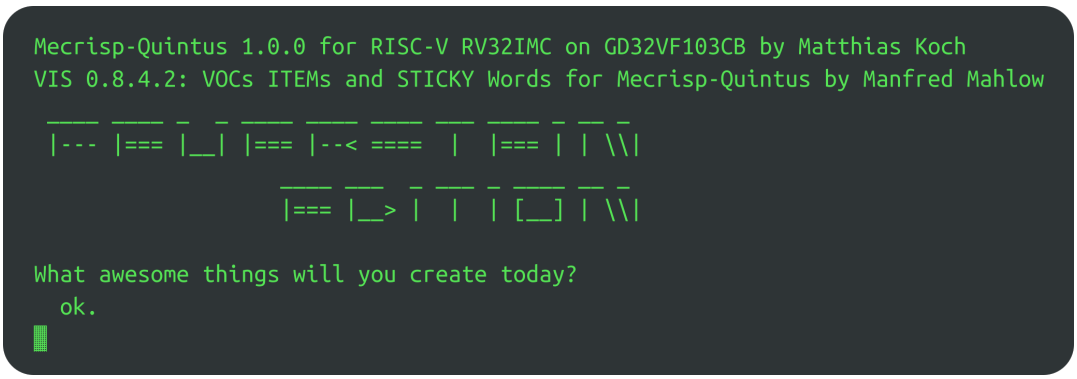


Abbildung 5: So meldet sich die Feuerstein–Variante von Mecrisp–Quintus

Einplatinencomputer

Rafael Deliano

Nicht nur für Breadboards haben sich steckbare Controller–Baugruppen als nützlich erwiesen. SBCs[1] *werden reichlich als handelsübliche Module angeboten. Wenn man hohe Rechenleistung, d. h. geschrumpfte PCs, wirklich benötigt, ist die Eigenentwicklung nicht angebracht. Oft genügt aber ein 8–Bit–Controller. Dann ist es effektiver, sich das zu bauen, was man haben will. Wenn man dem Keep–it–simple–Prinzip folgt, kommt auf die Platine wirklich nur die minimale Beschaltung, die man immer benötigt. Hier schildere ich meine Entwicklung der letzten Jahre.*

Ausgangslage

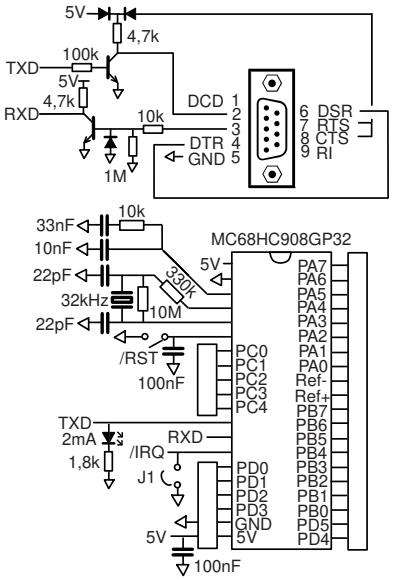


Abbildung 1: Schaltung GP32–SBC

Die Grundfunktion war hier: RS232 als Debugport, Betriebsanzeige–LED, Resettaster und der Quarz (Abb. 1). Das ergibt eine ein– oder zweilagige Leiterplatte geringer Komplexität, die noch gut von Hand bestückbar ist. Der Markt für Controller verändert sich rasch und für manche Applikationen passt die simple erste Platine nicht mehr. Das erzwingt dann ein neues Board, das aber kompatibel sein sollte.

Upgrade

Erst kam ein DIL40–Controller zum Einsatz (Abb. 2). Der ist heute nur noch in SMD–Bauform QFP44 beschaffbar und eine neue Platine war damit nötig (Abb. 3). Der eingelötete Controller ist durch diese Träger–Platine weiterhin wie ein DIL40–IC per Adapter steckbar im Programmiergerät. Zudem wurde die Gelegenheit genutzt, weitere Pins herauszuführen.

512 Byte RAM und 32 kByte Flash des GP32 reichten manchmal nicht aus. Der MC9S08AW60 hat mit 2 kByte SRAM und 60 kByte Flash mehr Speicher und ist auch doppelt so schnell. Er ist inzwischen genauso obsolet wie der GP32. Aber der wird rar und ist mit 8 EUR bei ebay.com unattraktiv teuer. Der neuere Automotive–Typ AW60 hatte mehr Verbreitung und ist mit 3,60 EUR billig und reichlich verfügbar. Durch gleiche CPU und 5–V–Versorgung sind beide kompatibel. Es wurde sorgfältig darauf geachtet, dass A/D–Wandler und SPI auf den gleichen Pins liegen (Abb. 4). Begrenzt durch die 40 Pins der Platine werden bei ihm nicht alle Portpins genutzt.

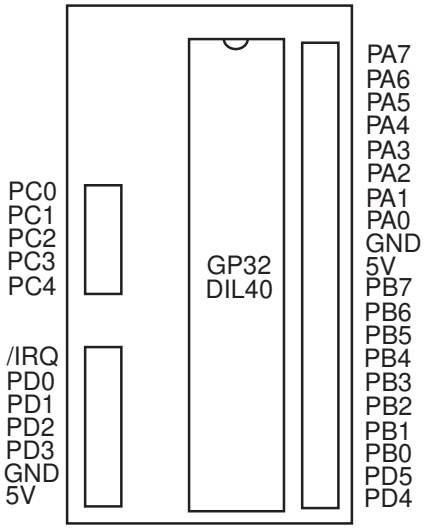


Abbildung 2: GP32–SBC DIL40

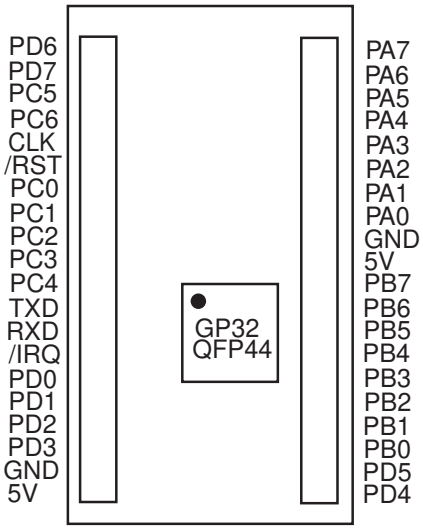


Abbildung 3: GP32–SBC QFP

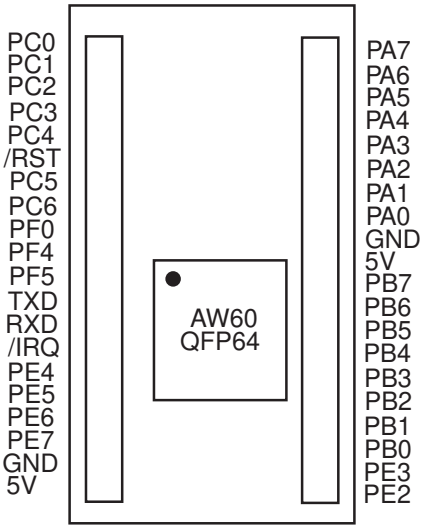


Abbildung 4: AW60–SBC

[1] Single Board Computer.

FORTH System Overview 2022

Name	Autor	Jahr	Last Version	Standard	ARM			x86	RISC-V	MIPS	PowerPC	8048	x51	8080/Z80	65xx	68xx	68xxx	Coldfire	AVR	PIC	MSP430	Others	OS					System	Bits	Description
					Cortex-M	Cortex-A	Cortex-R																CP/M	DOS	Windows	Linux	Other			
2k1Forth	D. K. Elvey	1997																				x								
44	Larry Battraw	1996		Small Forth							x																	PPC403	16	Small Forth-like system
4th	J.L. Bezemer	2016	3.62.4	ANS Standard																		x		x	x	x	x	different	16/32	Forth Compiler for different systems (using C or Pascal Comp.)
4THMACS	Hanno Schwalm	1994	3.0																			x						Archimedes RISCOS	32	Hanno Schwalm's port of Mitch Bradleys "Forthmacs 3.0".
51FORTH	Scott Gehmlich	1993											x															80C31	16	Only Forth assembler listing
64forth Cartridge	Tom Zimmer	2002		F83											x												x	C64	16	64FORTH is a cartridge program for the Commodore 64
6502FORTH	Programma International, Inc.	1979	1.2	FIG											x													6502	16	Only Users Manual available
75Forth	John Cassady	1983																				x						HP-75C	16	Forth for HP-75C Calculator
79-Forth	C. H. Ting	1983	1	FORTH-79																							x	Apple-II (6502)	16	Apple ROM Version with book "FORTH for the Complete Idiot"
80C552 FORTH System	Alberto Pasquale	1991											x															80C552	16	using CROS-16 (Universal Cross-Assemblers)
A4TH	Peter Appelman			Forth-83													x											Amiga1000	32	A Public Domain Forth system for Amiga's
AmForth	Matthias Trute	2020	6.9																x		x							MSP430, AVR	16	Extendible Command Interpreter
AmForth	Matthias Trute	2020	6.9		x				x																			LM4F, Hifive1	32	Still experimental
amrFORTH	A. L. Mitchell, C. Shattuck		V5										x		x							x		x	x	x		68HC11, C166	16/32	Use F-PC or Win32Forth
Arduino AVM	Mikael Patel	2016																	x									Arduino	16	Arduino Forth Virtual Machine
asforth	Jeff Doyle	2017																	x									Atmega328	16	Subroutine Threaded version of amForth
Atari Coin Op Forth	B. A. Miller, D. McIntyre, E. A. Klotz	1983		FIG											x													Atari 400/800	16	Variant of FIG Forth
ATLAST	John Walker		V1.0	FORTH-83													x							x				Atari ST, PC	16	Autodesk Threaded Language Application System Toolkit
Avice	cinetix@t-online.de	2016																	x									ATMega8/168 boards	16	AVR Virtual Stack Engine
BBL								x																				MS-DOS	32	BBL = Big Business Language or Big Black Lady (PD Forth)
bigForth	Bernd Paysan	2010	2.4	ANS Standard																					x	x		x86	32	32-bit Forth with tools like MINOS (GUI for X)
Blazin' Forth	Scott Ballantyne			F83											x												x	C64	16	A complete Forth-83 system for the C64
botForth	Steve Sheppard			Forth-83													x										x	68000	32	68000 Forth in assembler
ByteForth	Dutch Forth Group (HCC)		2.07/2.00										x						x									AVR, 8051	8	Supports different versions of AVR and 8051
Camel-Forth (new)	Bradford J. Rodriguez			ANS Standard	x									x	x													65C02, Rpi Pico, eZ80	16-48	Newer Versions
Camel-Forth 1802	Bradford J. Rodriguez	2009		ANS Standard																		x						RCA1802	16	Tested on Simulator
Camel-Forth 6809	Bradford J. Rodriguez	1995		ANS Standard												x												6809	16	Tested on ScroungeMaster II, MAIS AN601 port from Albert Nijhof
Camel-Forth 8051	Bradford J. Rodriguez	1999	1.6	ANS Standard									x															8051	16	Tested on Simulator
Camel-Forth 8086	Bradford J. Rodriguez	2009	0.2	ANS Standard				x																x				8086	16	DOS-Version
Camel-Forth CP/M	Bradford J. Rodriguez	1995		ANS Standard										x		x							x					CP/M (Z80)	16	Tested under CP/M
Camel-Forth MSP430	Bradford J. Rodriguez			ANS Standard																	x							MSP430	16	4e4th recommended as development environment
ciforth (lina)	Albert van der Horst (HCC)	2010	4.1	ISO Forth				x																x	x		x	Intel MAC	16/32	Same source for all implementations
cmForth	Charles H. Moore			(Moore)																		x					x	NC4000, RTX2000	16	Forth for FORTH-Chip Novics NC4000 and RTX-2000
coldForth	Colour Vision Systems	2001	0.8															x										Coldfire 5407	32	With TCP/IP - Telnet support
colorForth DOS	Charles H. Moore			(Moore)				x																x				DOS (386)	32	Type are in the color of each word
colorForth F18	Charles H. Moore			(Moore)																		x						F18	16	Type are in the color of each word
colorForth GA144	Charles H. Moore			(Moore)																		x						GA144	16	144 small Forth-Processors
colorForth Windows	Charles H. Moore			(Moore)				x																	x			Windows (386)	32	Type are in the color of each word
comFORTH	Egmond Woitzel			Forth-83				x																	x			Windows	32	Commercial Forth-System for Windows
coreForth	Eckhart Koeppen	2015		diverse	x																							Cortex-M0/M3	32	LM3S811, Arduino Due, Olimex STM32-P103 …
DCPU																						x						DCPU-16 (FPGA)		Forth for Notch's DCPU-16
DurexFORTH	Jkotlinski		1.3												x												x	C64	16	Modern C64 Forth with graphics and music support
DX-Forth	Ed ?	2001	3.98	Forth-83										x									x					CP/M-80	16	Experimental Forth Compiler for CP/M-80
DX-Forth	Ed ?	2001	4.17	Forth-83				x																x				DOS	16	Experimental Forth Compiler for DOS
eForth 56002	C. H. Ting, Dave Taliaferro			eForth																		x						DSP56002	16	New Model with ~200 Words often compiled with MASM
eForth 68000	C. H. Ting, Richard H. Haskell			eForth													x											68000	32	New Model with ~200 Words often compiled with MASM
eForth 6805	C. H. Ting			eForth												x												6805	16	Used for FPGA implementation
eForth 6811	C. H. Ting, Karl Lunt, W. Schemert			eForth												x												6811	16	New Model with ~200 Words often compiled with MASM
eForth 6812	C. H. Ting			eForth												x												6812	16	Optimized using subroutine threading
eForth 6816	C. H. Ting, Pete Zawasky			eForth												x												6816	16	New Model with ~200 Words often compiled with MASM
eForth 78C10	C. H. Ting			eForth																		x						78C10	16	New Model with ~200 Words often compiled with MASM
eForth 8051/XA	C. H. Ting			eForth									x															8051	16	New Model with ~200 Words often compiled with MASM
eForth 8080	C. H. Ting			eForth										x										x				8080 on FPGA	16	New Model with ~200 Words often compiled with MASM
eForth 8086/80386	C. H. Ting			eForth				x																x				DOS (8086)	16/32	New Model with ~200 Words often compiled with MASM
eForth 8098	C. H. Ting			eForth																		x						8098	16	New Model with ~200 Words often compiled with MASM
eForth ADSP2181	C. H. Ting			eForth																		x						ADSP2181	16	16-bit DSP version
eForth ARM7TMDI	C. H. Ting			eForth																		x						LPC2100, ADuC702x	32	eForth for ARM7TMDI
eForth ARM-M	C. H. Ting			eForth	x																							STM32F1/F4	32	eForth for Blue Pill
eForth AVR/Arduino	C. H. Ting			eForth															x									AVR328, Arduino	16	Direct implementation for AVR an C version for Arduino
eForth E220	C. H. Ting, Johannes Eißing			eForth																		x						Sharp E220 (Z80)	16	New Model with ~200 Words often compiled with MASM
eForth eP16/eP32	C. H. Ting			eForth																		x						FPGA	16	eP16 and eP32 in VHDL for Lattice XP2 Brevia Kit
eForth H8-532	C. H. Ting			eForth																		x						H8-532	16	New Model with ~200 Words often compiled with MASM
eForth in C	C. H. Ting			eForth																		x						different	16	Portables Forth for many systems

FORTH System Overview 2022

Name	Autor	Jahr	Last Version	Standard	ARM			x86	RISC-V	MIPS	PowerPC	8048	x51	8080/Z80	65xx	68xx	68xxx	Coldfire	AVR	PIC	MSP430	Others	OS					System	Bits	Description
					Cortex-M	Cortex-A	Cortex-R																CP/M	DOS	Windows	Linux	Other			
eForth in F#	C. H. Ting, Cheahsin Yap			eForth																		x						different	16	Windows XP
eForth in J1	C. H. Ting, Samawati			eForth																		x						different	16	J1 is a FPGA-Forth Processor
eForth in Java	C. H. Ting, Michael A. Losh			eForth																		x						different	16	Java Virtual Machine
eForth MIPS	C. H. Ting			eForth						x																		Silicon Graphics WS	16	Small C loader and use I/O routines from C library
eForth MSP430	C. H. Ting			eForth																	x							MSP430	16	New Model with ~200 Words often compiled with MASM
eForth MuP21	C. H. Ting		2.08	eForth																		x						MuP21	16	New Model with ~200 Words often compiled with MASM
eForth PDP1	C. H. Ting			eForth																		x						PDP1 on FPGA	16	New Model with ~200 Words often compiled with MASM
eForth PIC17	C. H. Ting			eForth																x								PIC17	16	New Model with ~200 Words often compiled with MASM
eForth PowerPC	C. H. Ting			eForth							x																	MPW on PowerPC-Mac	32	NEXT is a branch-through-link-register machine instruction
eForth STM8	C. H. Ting			eForth												x												STM8	16	STMicroelectronics STM8 is a modification of NXP HC08
eForth Transputer	C. H. Ting, Bob Barr			eForth																		x						Inmos Transputer	32	Experimental version
eForth Z80	C. H. Ting			eForth										x														Z80, Rabbit, E220	16	New Model with ~200 Words often compiled with MASM
Eulex	David Vázquez Púa	2009		Forth-79				x																				i386	32	A Bare Metal Forth Implementation
ESF Z-80 Forth	Vern Taliman	2011		FIG										x														Z80	16	Adaption of 8080 FIG Forth to Z80
F68K	Jörg Plewe	1992		Forth-83													x										x	68000-PC's	32	68000-Forth for Atari ST, Sinclair QL, Amiga and OS/9
F83 68000	Laxen & Perry	1983		Forth-83													x						x					CP/M 68000	32	Reference for many implementation
F83 8080/Z80	Laxen & Perry	1983		Forth-83										x														CP/M	16	Reference for many implementation
F83 8086	Laxen & Perry	1983		Forth-83				x															x	x				DOS, CP/M	16	Reference for many implementation
F83 Amiga	Laxen & Perry	1983		Forth-83													x											Amiga	32	Reference for many implementation
F83 Atari ST	Laxen & Perry	1983		Forth-83													x											Atari ST	32	Reference for many implementation
F83 NC4000	Laxen & Perry	1983		Forth-83																		x						NC4000	16	Reference for many implementation
F-TP	Fred Behringer	1996	1.00	F83, TurboForth																		x						T80x	16	Forth for the Transputer T80x
FANF	Konstantin Dimitrov	2014	1408																	x								PIC32MX170	32	Forth-link Programming Language for PIC32MX170
Fast 8 Forth	M. Simon	1996		FIG																		x						Z8, Super8	16	High speed Forth based on a modified version of fig-Forth model
FastForth MSP430	Jean-Michel Thorrens																				x							MSP430FRxxxx	16	For TI's LaunchPad MSP-EXP430FR[5969|6989|5994|5739|4133]
FCC	Braden Shepherdson	2020		ANS Standard		x		x																				x86 and ARMv7	32	FCC - Portable C engine
feline	Peter Graves	2021		Forth-2020				x																				AMD-64	32	64-bit native code Forth 200x
FF2-6800	Wilson M. Federici			FIG												x												6800	16	Forth83/Flex2 distribution disk
FF2-6809	Wilson M. Federici			FIG												x												6809	16	Forth83/Flex2 distribution disk
Fifth	CLICK Software							x																				DOS	16	Fifth is an interactive program development environment
FIG-Forth 1802				FIG																		x						RCA1802	16	Mostly Listing and source code available
FIG-Forth 386-DOS				FIG				x																x				DOS (386)	32	Mostly Listing and source code available
FIG-Forth 6502				FIG											x													6502	16	Variants also for Atari 400/800, Apple II, Nova, and Eclise C
FIG-Forth 6800				FIG												x												6800	16	Mostly Listing and source code available
FIG-Forth 68000				FIG													x											68000	32	Mostly Listing and source code available
FIG-Forth 68020				FIG													x											68020	32	Forth for NEC Astra
FIG-Forth 6809				FIG												x												6809	16	Mostly Listing and source code available
FIG-Forth 6811				FIG												x												6811	16	Mostly Listing and source code available
FIG-Forth 8080				FIG										x														8080, TRS-80	16	Mostly Listing and source code available
FIG-Forth 8086-DOS				FIG				x																x				DOS (8086)	16	Mostly Listing and source code available
FIG-Forth 8086-IBM				FIG				x																x				DOS (IBM 8086)	16	Mostly Listing and source code available
FIG-Forth 9900				FIG																		x						TI 99/4	16	Mostly Listing and source code available
FIG-Forth Alpha				FIG																		x						Alpha Micro	16	Mostly Listing and source code available
FIG-Forth Apple-II				FIG											x												x	Apple-II (6502)	16	Mostly Listing and source code available
FIG-Forth Eclipse C				FIG																		x						Eclipse C	16	Mostly Listing and source code available
FIG-Forth Nova				FIG																		x						Nova	16	Mostly Listing and source code available
FIG-Forth PACE				FIG																		x						National PACE	16	Mostly Listing and source code available
FIG-Forth PDP11				FIG																		x						PDP11	16	Mostly Listing and source code available
FIG-Forth TRS-80				FIG										x													x	TRS-80 (8080)	16	Mostly Listing and source code available
FIG-Forth VAX				FIG																		x					x	VAX (Unix)	16	Mostly Listing and source code available
FIG-Forth Z80				FIG										x													x	Z80, CP/M-80, Amstrat	16	CP/M-80, Amstrat CPC464
FIRST	W.Schemmert	1994		ANS Subset				x																x				DOS	16	FIRST is an ANS Forth Standard CORE subset
FISH Forth	M. L. Simon, C. W. Phillips Jr.			FIG	x																							LPC8xx/11xx, STM32	32	LPC812-, LPC111x-, and STM32F4 Discovery Board
FlashForth	Mikael Nordman																		x	x								PIC 18/24, dsPIC, Atmega	16	Standalone native Forth operating system
FOeRTHchen	Helmar Wodtke			Subset																								x86, Javascript, Perl	32	The small Forth
ForST	John Redmond	1991															x											Atari ST	32	68K subroutine-threaded Forth with optimisation
Forth II	William G. Graves														x												x	Apple-II (6502)	16	Apple ROM Version with book "FORTH for the Complete Idiot"
Forth09	D. P. Johnson	1988		Forth-83												x												OS-9 for 6809	16	Forth for OS-9 running on 6809
Forth-11	Thomas E. McGuire																					x						PDP11	16	Kitt Peak Multi-Tasking Forth
FORTH2020	Peter Forth	2020		ANS Standard																					x			Windows	32	Based on WinForth
FORTH51	William H. Payne												x															8051	16	Book available: William H. Payne - Embedded Controller FORTH
FORTH68K	Andy Valencia	1994		Forth-83													x										x	68000	32	68000 assembler source
FORTH86	Jerry Boutelle			FIG				x																				DOS	16	Adaption of FIG-8086 to NAUTILUS metacompiler

FORTH System Overview 2022

Name	Autor	Jahr	Last Version	Standard	ARM			x86	RISC-V	MIPS	PowerPC	8048	x51	8080/Z80	65xx	68xx	68xxx	Coldfire	AVR	PIC	MSP430	Others	OS					System	Bits	Description
					Cortex-M	Cortex-A	Cortex-R																CP/M	DOS	Windows	Linux	Other			
ForthARM	Braden Shepherdson	2020		ANS Standard		x																						ARMv6	32	Forth for Raspberry PI
ForthCPM	Tom Almy	2018		ANS or FIG				x																				DOS	16	Compiles Forth into machine code
FORTHdsPIC	Bill Marshall ?	2013	0.5																	x								PIC24/dsPIC	16	Examples for FFT and other bare metals
F-PC	Tom Zimmer	1991	3.56	Forth-83				x																x				DOS	32	Standard for DOS Forth
Gforth EC				ANS Standard	x			x		x					x		x					x						H8, R8C, C166, NXT …	16	Embedded Gforth version for Lego NXT and much more
Gforth Linux	Crook/Ertl/Kühhling/Paysan/Wilke		0.7.9	Forth-2020		x		x	x	x	x															x		Linux	32/64	Portables Forth for many systems
Gforth Win32	Crook/Ertl/Kühhling/Paysan/Wilke		0.7.9	Forth-2020				x																	x			Windows (386)	32	Portables Forth for many systems
Gforth Win64	Crook/Ertl/Kühhling/Paysan/Wilke		0.7.9	Forth-2020				x																	x			Windows (x64)	64	Portables Forth for many systems
GraForth	Paul Lutus														x											x		Apple-II (6502)	16	Forth with graphic and sound support
GSForth															x											x		Apple-II (6502)	16	No information - only disc images
hForth	Wonyong Koh			eForth		x		x						x														StrongArm, x86, Z80	16/32	Based on eForth
Holon	Wolf Wejgaard	1996	4.0					x																x				DOS	16	Modular Forth
IsoMax																x												6811	16	Forth-like Language for RT-application and Isostructure
Java Forth	Frank Buss			Java Forth																		x					x	Web-Browser	32	Class-Implementation of Forth
Jax4th	Jack J. Woehr	2000	1.25/0.3.17					x									x											Windows NT, Amiga	32	Free Forth written in MASM6.11
Jforth	Mike Haas, Phil Burk	1998		Forth-83													x											Amiga	32	JForth PD is a complete JForth software development system
Jupiter ACE Forth															x													Jupiter ACE	16	One of few personal computer with Forth in ROM
Kevo	Antero Taivalsaari	1992	0.9b2														x											Apple Macintosh	32	
kforth (32/64)	Krishna Myneni	1998						x																		x	x	Linux (386/586) / Windows	32/64	32- or 64-bit Forth System for x86-GNU/Linux
KK-Forth 68332	Klaus Kohl	1990	1.2	Forth-83													x											68332	16	68332 version (16-Bit RAM version)
KK-Forth 8415	Klaus Kohl	1990	1.2	Forth-83										x														84C015 (Z80)	16	EMUF-84C015 version
KK-Forth C165	Klaus Kohl	1998	1.3.1	Forth-83																		x						C165	16	Infineon C165 version
KK-Forth FG	Klaus Kohl	1990	1.2	Forth-83																		x						NC4000	16	Novics NC4000 version
KK-Forth PC	Klaus Kohl	1990	1.2	Forth-83				x																x				DOS (8086)	16	DOS implementation with terminal for other versions
KK-Forth V20	Klaus Kohl	1990	1.2	Forth-83				x																				V20 (8086)	16	Version for NEC V20 (8086 compatible)
L.O.V.E Forth	H. Seywerd, W. R. Elehew, P. Cavén	1988		Forth-83				x																				DOS	16	Distributed under the User Supported software concept
LaForth	LaFarr Stuart, Robert L. Smith			FIG				x																x				DOS	16	An Experimental Forth For IBM-PC
LMI Forth	Ray Duncan, A. Flesch	1987		Forth-83												x												6811	16	Source and Assembler
LMI UR/Forth	Laboratory Microsystems	1987		Forth-83				x																				DOS (8086)	16	Commercial Forth-System for DOS
lpforth	Jih-tung Pai	2001						x																		x		Linux	32	Small Forth for Linux
MaxForth	MS Microscan	1986														x												F68HC11	16	6811 with Forth in ROM (New Micros NMIS-0021 Forth SBC)
Max-Forth	MS Microscan			Forth-83												x												6812	16	MAX-Forth adaptet to 68HC12
mcForth	Klaus Kohl-Schoepe	2020	1.0	ANS Standard	x																	x			x			XMC1100, Windows	32	Cortex-M0 for XMC1100, Virtual Processor in MASM for PC
mecrisp EFM32	Matthias Koch			ANS Standard	x																							EFM32GG/HG	32	Some Silicon Labs EFM32
mecrisp J1A	Matthias Koch			ANS Standard																		x						J1A (FPGA)	32	e.g. for Intel MAX10
mecrisp Kinetis K	Matthias Koch			ANS Standard	x																							K20	32	Some NXP Kinetis K
mecrisp Kinetis L	Matthias Koch			ANS Standard	x																							KL25/46	32	Some NXP Kinetis L
mecrisp LPC111x	Matthias Koch			ANS Standard	x																							LPC111x	32	LPC111x-Family of Cortex-M0
mecrisp MSP430	Matthias Koch			ANS Standard																	x							MSP430	16	MSP430 Version
mecrisp MSP432	Matthias Koch			ANS Standard	x																							MSP432	32	Cortex-M4F with MSP430 peripheral
mecrisp NRF	Matthias Koch			ANS Standard	x																							NRF51822	32	Nordic-Version
mecrisp RISC-V	Matthias Koch			ANS Standard					x																			GD32VF103 …	32	Different variants
mecrisp Stellaris	Matthias Koch		2.2.4	ANS Standard	x																							LM3S/LM4F	32	Many TI Stellaris MF3S and LM4F120 (now TM4C)
mecrisp STM32	Matthias Koch			ANS Standard	x																							STM32F1/2/3/4	32	Nearly complete STM32F-Family
mecrisp XMC1100	Matthias Koch			ANS Standard	x																							XMC1100	32	Infineon XMC1xxx-Family
megaforth (genforth)	Erik Haliewicz																x											68000	32	Forth designed for the Sega Megadrive
microForth	Forth Inc.	2009		ANS Standard																		x						RCA1802	16	Forth for RCA COSMAC
minForth	John Williamson			ANS Standard		x																		x				ARMv7	32	Bare-metal Raspberry Pi port
MinForth DOS	Andreas Kochenburger			ANS Standard				x																x				DOS (8086)	16	Minimalistic but complete FORTH System in (Turbo-)C
MinForth Linux	Andreas Kochenburger			ANS Standard				x																		x		Linux (586)	32	Minimalistic but complete FORTH System in (Turbo-)C
MinForth Win32	Andreas Kochenburger			ANS Standard				x																	x			Windows (586)	32	Minimalistic but complete FORTH System in (Turbo-)C
MINI4TH	Ted Beach	1988		Small Forth				x																x				DOS	16	Small Forth system based on ZEN Forth
Moonforth																						x						DCPU-16 (FPGA)	16	Forth for the FPGA implementation of DCPU-16
MOPS	Michael Hore	1995	2.6														x											Apple Macintosh	32	
MPE MSP430 Lite	Microprocessor Engineering Limited	2013	v7.3																		x							MSP430	16	Lite version of commercial Forth
muforth	muforth.nimblemachines.com			Small Forth	x				x						x					x	x							AVR, HC08, PIC18 …	16	Small, simple, fast, indirect-threaded code Forth
MyForth	Charley Shattuck, Bob Nash	2006											x						x									805x, ADuC841, AVR	16	A Minimalist 8-Bit Forth using ideas from Colorforth
nanoFORTH	Rafael Deliano	2004		Forth-83												x												68HC908GP32	16	Available with assembler
Ninth Forth	YakPeople - Strick Yak	2016														x											x	6809 (OS-9)	16	Forth
nop	https://github.com/iru-/nopforth							x																		x		Linux, macOS, Windows	64	Nop is a dialect of the Forth programming language for x86_64
OF816	https://github.com/mgcaret	2020		IEEE 1275-1994											x													65C816	16	65C816 Forth heavily inspired by Open Firmware
OneForth	John Redmond	2015		F83													x											Atari ST	32	Supports all GEMDOS functions
Open Firmware	Mitch Bradley			IEEE 1275-1994		x				x	x																x	different	32	Bootloader for many high end systems

FORTH System Overview 2022

Name	Autor	Jahr	Last Version	Standard	ARM			x86	RISC-V	MIPS	PowerPC	8048	x51	8080/Z80	65xx	68xx	68xxx	Coldfire	AVR	PIC	MSP430	Others	OS					System	Bits	Description
					Cortex-M	Cortex-A	Cortex-R																	CP/M	DOS	Windows	Linux	Other			
ORForth	Robert Edwards	2008		Forth-83																		x						Z8	16	using 32K EPROM and 32K RAM
OSX-Forth	Chris Hinsley	2015						x							x													AMD-64	32	Forth for OSX, bootstrap from x86 NASM then Forth only
Pettil															x													PET2001 (6502)	16	
pijForthos	Dale Schumacher, Tristan Slominski	2014				x																						Raspberry Pi	32	Raspberry Pi JonesFORTH O/S
ppForth	Jih-tung Pai	1991	08102001																			x						Palm OS	32	Palm Pilot Forth
preForth / seedForth	Ulrich Hoffmann			Minimal				x																				i386	32	Only 13 primitives - seedForth complete it to full Forth
PropellerForth	Cliff L. Biffle	2006																				x						Propeller	16	For Parallax Demo Board
PropForth	Sal Sanci																					x						Propeller	16	An Introduction to Interactively programming parallel processors
Punyforth	Attila Magyar	2018		Forth inspired ...		x		x														x					x	Rpi, ESP8266	32	Simple stack-based Language for IoT
Pygmy	Frank Sergeant	1990	1.7	Small Forth/ANS				x																x				DOS	16	Small Reference to ANS
Qforth	Toshiyasu Morita, R. Kneusel														x												x	Apple-II (6502)	16	A simple 16-bit integer Forth for ProDOS
RAFOS FORTH	Everett Carter	1986	V1.0	Forth-79												x												6805	16	
Reva	Ron Aaron			ANS Standard				x																		x	x	Windows, Linux	32	Based on RetroForth with ideas from HelFORTH
rpiForth	https://github.com/aviatorRHK/rpiForth	2013				x																						Raspberry Pi	32	A Forth Operating System for Raspberry Pi
RSC Forth	Rockwell International	1983	1.7	FIG											x													R65F11/12	16	FORTH im ROM / Rockwell R65FR1 Forth Develoment ROM
SectorForth	Cesar Blum	2020		Minimal				x																x				x86-Bootversion	16	A 512-byte, bootable x86 Forth
SFORTH	Alan DeMars			Forth-83													x										x	68000	32	68000 assembler source
SixtyForth	David Jones	2018		ANS Standard				x																				AMD-64	32	64-bit Intel assembly
STM8EF	Thomas (TG9541)			eForth												x												STM8	16	Based on eForth STM8 using SDCC
StrongForth	Dr. Stephan Becher			ANS Standard				x																				DOS	16	ANS Forth with strong type checker
Super8 Forth	Klaus Kohl	1991		Forth-83																		x						Super8	16	ROM-Version of Zilog Super8
SwapForth FT900	James Bowman			ANS Standard																		x						FT900	16/32	Cross-Platform 16-/32-Bit ANS-Forth for FDTI FT900
SwapForth J1a	James Bowman			ANS Standard																		x						J1A (FPGA)	16/32	Cross-Platform 16-/32-Bit ANS-Forth for J1A
SwiftX MSP430	Forth Inc.																				x							MSP430	16	SwiftX Integrated Cross-Development Software
Tacyon	Peter Jakacki	2012	3.0	ANS Standard																		x						Propeller	16	Fast Forth for Parallax Propeller with FAT32+FGA
Tali Forth	Scot W. Stevenson	2014													x													65C02	16	STC-Implementation of ANS
tForth	Dutch Forth Group (HCC)	1991																				x						Transputer	16	ANSI Forth for the transputer with parallel facilities (Commercial)
The Virtuall Forth Computer				FIG																		x						PDP11	16	FIG-Forth adaption to PDP11
Tiny Open Firmware	Brad Eckert		3.03	IEEE 1275-1994									x				x	x	x									Coldfire, 68K, AVR, x51	16	Based on Open Firmware (use Fcode)
TurboForth	Mark Wills																					x						TI-99/4A	16	TMS9900 Forth for Texas Instruments TI-99/4A
ucForth																		x								uC		Coldfire 5272	32	Using ucLinux on Coldfire
UniForth 6502	Unified Software Systems														x												x	Apple-II (6502)	16	Subset of professional UNIFORTH (UniForth Sampler)
UniForth 68000	Unified Software Systems			Forth-83													x										x	68000	32	Subset of professional UNIFORTH (UniForth Sampler)
UniForth Z80	Unified Software Systems	1985		Forth-83										x													x	Z80	32	Subset of professional UNIFORTH (UniForth Sampler)
ValForth	Valpar International	1982		FIG											x													Atari 400/800	16	Support for graphic games
VIC Forth	Tom Zimmer	1982													x												x	Commodore VIC-20	16	Forth cartridge for Commodore VIC-20
volksForth Apple-II			3.8	Forth-83											x												x	Apple-II (6502)	16	Forth-83 Implementation by Forth e.V.
volksForth Atari 400			3.81.03	Forth-83											x												x	Atari-8 Bit (6502)	16	Forth-83 Implementation by Forth e.V.
volksForth Atari ST			3.81	Forth-83													x										x	Atari ST (6502)	32	Forth-83 Implementation by Forth e.V.
volksForth C64			3.81	Forth-83											x												x	Commodore C64 (6510)	16	Forth-83 Implementation by Forth e.V.
volksForth CPM			3.8	Forth-83										x									x					Schneider CPC (CP/M)	16	Forth-83 Implementation by Forth e.V.
volksForth DOS			3.81.41	Forth-83				x																x				DOS	16	Forth-83 Implementation by Forth e.V.
volksForth Z80			3.80	Forth-83										x									x					CP/M-2.2 (Z80)	16	Forth-83 Implementation by Forth e.V.
WHYP	Richard E. Haskell															x	x											68HC11/12, 68332	16/32	WHYP = Words to Help You Program
Win32Forth	Tom Zimmer and many more		6.14.04	Full Version				x																	x			WIN32	32	Full blown Windows System
YERC	Kriya Systems, Inc.																x											Apple Macintosh	32	Yerk is an object oriented language based on a Forth Kernel
Z8001 FORTH	Lou Odette	2005												x														Z8000/Z8001	16	Adapted from the Dr. Dobbs No. 71 (1982)
ZEN Forth	Martin Tracy			Small Forth				x																x				DOS	16	Small Reference to ANS
Zforth	Impossible Mission Force	2019		Forth-79												x												6809 (HD63C09)	16	For Hitachi HD63C09 single board computer
Zimmer Forth	Tom Zimmer			Forth-83				x																x				DOS	16	Used for some trainings
ZX81-Forth	Skywave													x										x				ZX-81	16	Forth for ZX-81 with at least 16K RAM

Downgrade

Eine kurze Variante der GP32–QFP44–Trägerplatine verzichtet auf LED, Taster und RS232–Stecker, weil diese oft in Geräten in der Frontplatte vorhanden sind. Dafür braucht es dann aber eine kompaktere, flache Bauform. Es wurden hier keine neuen PCBs gefertigt, die Platinen wurden schlicht abgesägt (Abb. 5).

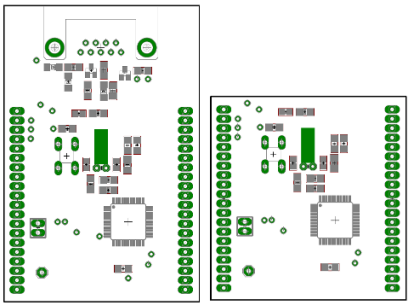


Abbildung 5: Gekürzte Variante GP32 QFP

Wie sich eine SBC–Familie im Laufe der Jahre entwickelt, ist schwer vorhersehbar und planbar. Neben Verzicht auf unnötige Schaltungsteile und Wahl der Versorgungsspannung ist vor allem eine solide Mechanik wichtig (Abb. 6).

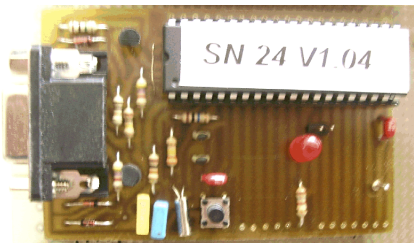


Abbildung 6: Erster Prototyp GP32–SBC DIL40 auf einlagiger Leiterplatte

0,635–mm–Vierkantstifte sind robuster als Stifte für IC–Kontakte. Ihre Buchsenleisten mit gedrehten Kontakten überstehen viele Steckzyklen (Abb. 7), selbst wenn mal unsanft leicht verkantet montiert wird.

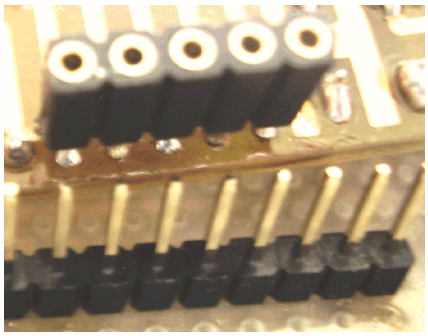


Abbildung 7: 0,635–mm–Buchsen– und Stiftleisten

Bei etwa 40 Pins und reichlichen Abmessungen ist das Board bereits gut fixiert. Mechanische Belastung ergibt sich besonders durch seitliche Steckverbinder. Bei kleinen Boards können Schrauben helfen, diese am Platz zu halten (Abb. 8).

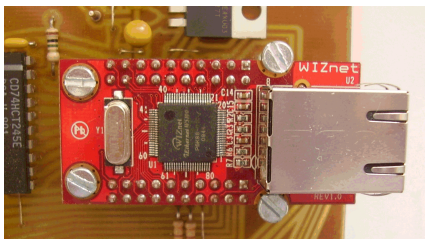


Abbildung 8: Schrumpf–Board von WIZnet mit Ethernet–Buchse und M3–Befestigungsschrauben

Als Option sind die auch bei größeren Boards manchmal wünschenswert. Raster und Position der Steckverbinder muss 2,54 mm betragen, sonst passen sie nicht für Lochrasterplatinen. Selbst wenn man diese Anforderung nicht hat, ein Übergang von 2,54 mm auf 2,00 mm spart nur geringfügig Platz, aber die Auswahl an verfügbaren Bauformen der Stecker sinkt drastisch. Die Abmessungen der Leiterplatte sollte etwas Spielraum für zukünftige Erweiterungen lassen. Durch den Wechsel von bedrahteten Bauteilen auf SMD (Abb. 2, 3) wurde hier weiterer Platz verfügbar. Bei der Wahl der SMD–Bauteile sollte man berücksichtigen, ob sie sich für die Handbestückung eignen. Fine Pitch[2] sorgt oft für mehr Probleme, als sich Vorteile durch die höhere Packungsdichte ergeben (Abb. 9, 10). Eine Entscheidung, die besonders gut überlegt sein will, ist die Versorgung des Controllers. Und auch, mit welchen Pegeln die Portpins arbeiten. Alle diese Boards können mit 3,3 V . . . 5 V betrieben werden. Der ursprüngliche GP32 ist so alt, dass das für ihn normal war. Der AW60 als Automotive–Controller hat intern niedrige Core–Spannung, verhält sich aber nach außen wie ein 5–V–Controller. Es ergaben sich nur wenige Anwendungen, die mit 3,3 V betrieben wurden, typischerweise die Ansteuerung von Speichern. Alles was analog ist, lässt sich leichter mit 5 V bauen. Bei 8–Bit–Controllern ist diese Spannung recht üblich. Es gibt aber auch eine kleine Auswahl ARM–M0 mit 2,7 V . . . 5 V Versorgung. Das sind nicht die billigsten und gängigsten ICs, aber sie bieten Flexibilität in der Beschaltung. Grundsätzliches Problem aller ARM–Controller ist jedoch, dass sie sich nicht so gut für eine Bit–Banger–Software eignen wie die 8–Bit–Controller.

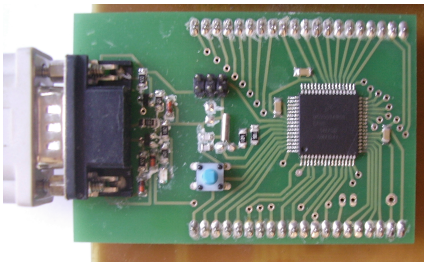


Abbildung 9: AW60–SBC

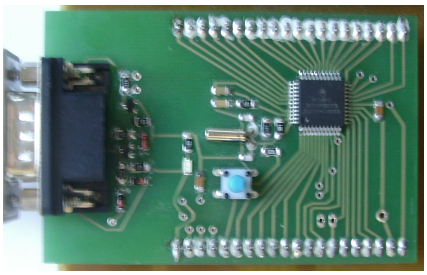


Abbildung 10: GP32–SBC QFP

Software

Nicht nur bei Testplatinen, auch bei den meisten Seriengeräten bestimmt

[2] Als „Pitch“ wird bei elektronischen Bauteilen der mittlere Abstand der Anschlüsse zueinander bezeichnet. Ist dieser Abstand kleiner als 0,5 mm, spricht man von „Fine Pitch“.

der Zeitaufwand für Software die Entwicklungskosten. Also die gewohnte Umgebung, in der man am produktivsten arbeitet, nicht ohne zwingenden Grund wechseln! Zugekaufte SBCs sind nach einigen Jahren nicht mehr lieferbar. Bei selbstgebauten Platinen sind die Materialkosten gering. Zumindest für Anwendungen in Kleinserie kann man PCBs und Controller einlagern und nach 10 Jahren immer noch nachfertigen.

Breadboards

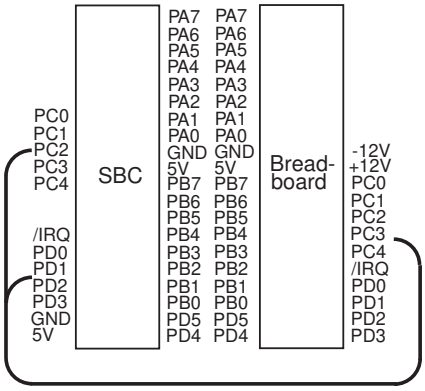


Abbildung 11: Breadboard–Adapter–Verdrahtung

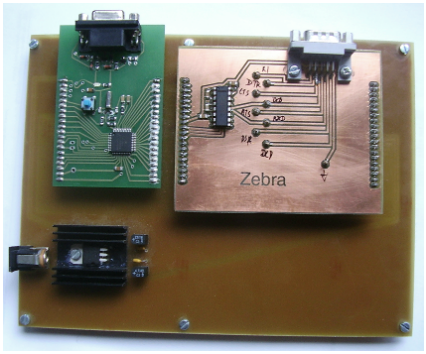


Abbildung 12: Breadboard–Adapter

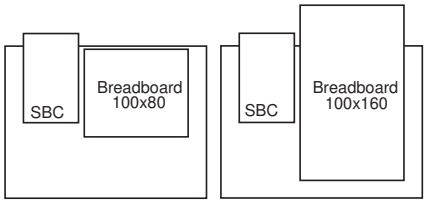


Abbildung 13: Varianten Breadboard

Wenn man Platinen ätzt, verwendet man meist das einheitliche Format der Europakarten (100x160 mm²). Es erwies sich, dass hauptsächlich kleine Testschaltungen benötigt werden, für die eine halbe Europakarte ausreichend ist. Eine sinnvolle Ergänzung zum SBC ist dann eine Platine (Abb. 12), die aus einem 12–V– oder 24–V–Steckernetzteil die 5 V erzeugt und die Verdrahtung zur Peripherie herstellt. Die Pinbelegung wählt man so, dass sie bequem auf einer einlagigen Platine layoutet werden kann (Abb. 11). In einigen Fällen erfordern umfangreiche Schaltungen volle Europakarten, die sind auch bestückbar (Abb. 13).

Anmerkung

Auf diesen Boards wird ausschließlich nanoFORTH verwendet. Die Variante für AW60 hat nur die IO angepasst und entspricht sonst der alten GP32–Version von 2004. Auf dem GP32 mit 512 Byte RAM funktioniert FORTH problemlos, aber C wäre wohl mühsam.

MyNOR Single Board Computer

Dennis Kuschel aus Bremen hat Wundervolles vollbracht: einen Einplatinen–Computer, aufgebaut um *ein einziges NOR–Gatter* aus zwei Transistoren, mit nichts als TTL–Gattern drum herum. Einige von euch kennen ihn vielleicht schon. Allen anderen sei seine Homepage ans Herz gelegt.

> „MyNOR is not just a fun project. It's a project that shows what can be done with just a few components of the good old days. It shows how a CPU works and that a CPU does not necessarily need to have an ALU. And it's the little brother of my other project MyCPU."

Und da kein Computer ohne Forth sein kann, voilà, Dennis hat auch das tatsächlich geschafft:

> „2022–05–24: I have written a Forth system for MyNOR and TraNOR. Are you interested in a very old programming language? Read more here: `http://mynor.org/forth.htm` ."

cas/mk

Im Web: `mynor.org` und `mycpu.eu`

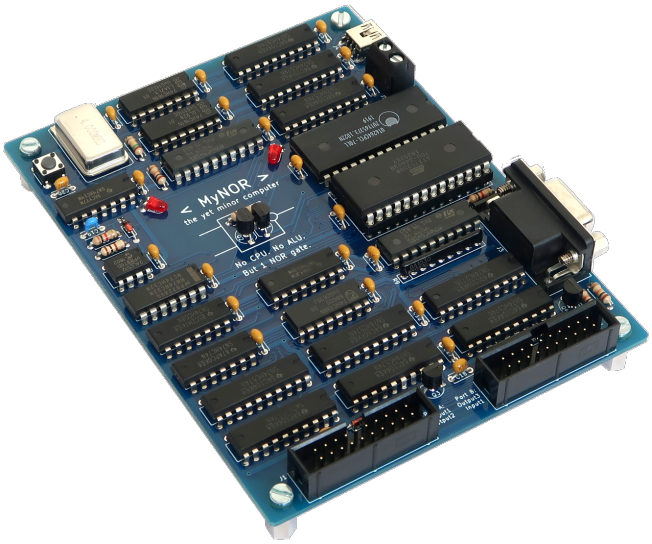


Abbildung 14: Die MyNOR–Computerplatine in ihrer ganzen Schönheit

Einfacher Selbsttest für SBC

Rafael Deliano

Nachdem man eine Leiterplatte bestückt hat, ist ein Endtest wünschenswert. Eine simple Schaltung mit kurzem Programm reicht aus, die Portpins zu überprüfen. Das ist nicht nur für die Fertigung nützlich. Auch auf Breadboards kommt es manchmal zum Kontakt eines Portpins mit Spannungen außerhalb von 0 V . . . 5 V, z. B. durch einen abrutschenden Testclip. Dann wird die Elektronik des Pins zerstört, aber der Controller bleibt ansonsten funktionsfähig. Solche teildefekten ICs will man schnell identifizieren.

Das Testprinzip

Es werden dabei jeweils zwei benachbarte Pins verbunden (Abb. 1). Die beiden Pins können sich wechselseitig als Ausgang high/low und Eingang testen (Abb. 2). Am angenehmsten, weil völlig schematisch testbar, sind Ports, an denen alle 8 Pins verfügbar sind. Sonderfälle machen den Test komplizierter. Der /IRQ–Pin kann z. B. zwar als Eingang high/low geschaltet werden, liefert als Ausgang aber durch seinen internen Pullup nur einen High–Pegel (Abb. 3).

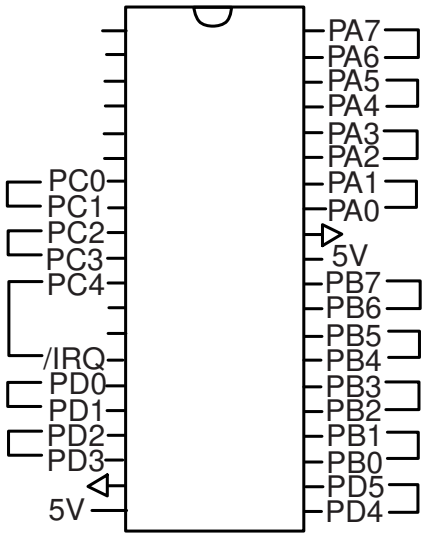


Abbildung 1: Verdrahtung für den GP32–DIL40–Tester

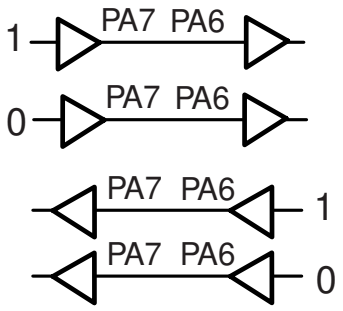


Abbildung 2: Testablauf an den Portpins

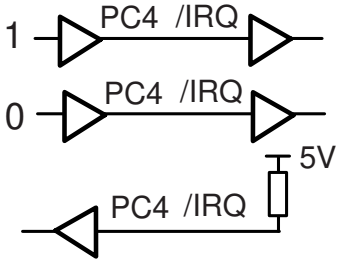


Abbildung 3: Sonderfall /IRQ

Programm

Kann man beim Laden über RS232 durch den Interpreter ausführen (s. Listing). Einige Befehle, wie hier den `IRQ@`, muss man jedoch vorher compilieren. Das ist bei Von–Neumann–CPUs wie dem 68HC908 kein Problem, da Programme im RAM ausführbar sind.

Hier wird bei Ausführung (Abb. 7) pro Zeile ein Port getestet und der Zustand der Pins als Bitmuster angezeigt. Am Ende der Zeile und am Ende des Programms wird ein Flag ausgedruckt, bei dem der Wert `00` anzeigt, dass der Test erfolgreich war.

Varianten

Ausgehend von der ursprünglichen DIL40–Grundversion (Abb. 1) sind im Laufe der Zeit drei im Pinout weitgehend kompatible Platinen entstanden. Auf der Testplatine sind deshalb versetzt angeordnete Stiftleisten vorhanden, damit man alle Boards testen kann (Abb. 4).

Bei der Short–Version (Abb. 5) ist der vordere Teil, der mit der Sub–D9–Buchse, abgesägt und die Resettaste und die LED fehlen. Diese Version wird nicht für Breadboards, sondern innerhalb von Geräten eingesetzt. Die nun extern vorhandenen Schaltungen (Abb. 6) sind auf der Unterseite des Testboards in SMD ausgeführt. Wenn lange Platinen gesteckt sind, sind diese Funktionen doppelt vorhanden und liegen parallel. Durch die Wired–OR–Schaltung von RXD ist das unproblematisch.

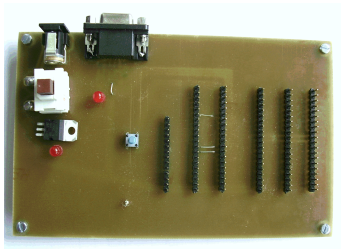


Abbildung 4: Testplatine mit den versetzten Stiftleisten

Für die 5 V ist ein großer, leicht bedienbarer Druckschalter angenehm, damit man die Versorgung während der Montage der Leiterplatten abschalten kann.

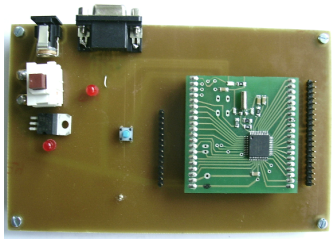


Abbildung 5: Bestückt mit „GP32 QFP short“

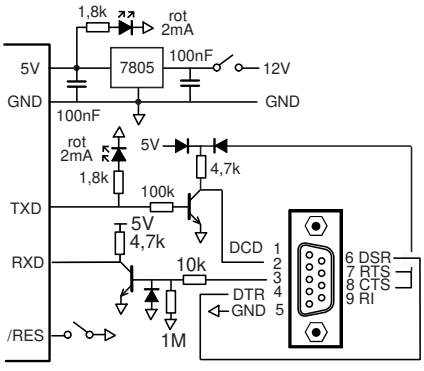


Abbildung 6: Stromversorgung und RS232 auf der Testplatine.

```
---- ---- ---- | <|
+ IRQ@
PA=11111111 00000000 11111111 00000000 00
PB=11111111 00000000 11111111 00000000 00
PC=00011111 FF 00010000 FF 00011111 FF 00000000 00 00
PD=00111111 00000000 00111111 00000000 00
00
RESET
---- ---- ---- |
|>
```

Abbildung 7: Screenshot eines Testlaufs.

Listing

```
1   Listing 1: Testprogramm für Variante SBC GP32 DIL40
2   <| \ DIL-GP32
3   1C0 >C !     \ compile to RAM
4   :CODE IRQ@   \ (   ---  Flag )
5   \ Read IRQ-Pin
6   \ Flag = 0000 , FFFF
7         A. CLR,
8   1 $       BIL,
9         A. DEC,
10  1 $:  DEX, 0,X STA,
11        DEX, 0,X STA,
12           RTS,
13  CODE;
14  0     \ OK-Flag
15  CR ." PA=" B% 10101010 DPA C! FF PA C! PA C@ DUP CB. NOT OR
16              B% 10101010 DPA C! 00 PA C! PA C@ DUP CB.     OR
17              B% 01010101 DPA C! FF PA C! PA C@ DUP CB. NOT OR
18              B% 01010101 DPA C! 00 PA C! PA C@ DUP CB.     OR DUP CH.
19  CR ." PB=" B% 10101010 DPB C! FF PB C! PB C@ DUP CB. NOT OR
20              B% 10101010 DPB C! 00 PB C! PB C@ DUP CB.     OR
21              B% 01010101 DPB C! FF PB C! PB C@ DUP CB. NOT OR
22              B% 01010101 DPB C! 00 PB C! PB C@ DUP CB.     OR DUP CH.
23  CR ." PC=" B% 10101010 DPC C! FF PC C! PC C@ 1F AND DUP CB. 1F XOR OR IRQ@ DUP CH. NOT OR
24              B% 10101010 DPC C! 00 PC C! PC C@ 1F AND DUP CB. 10 XOR OR IRQ@ DUP CH. NOT OR
25              B% 01010101 DPC C! FF PC C! PC C@ 1F AND DUP CB. 1F XOR OR IRQ@ DUP CH. NOT OR
26              B% 01010101 DPC C! 00 PC C! PC C@ 1F AND DUP CB.  OR IRQ@ DUP CH. OR DUP CH.
27  CR ." PD=" B% 10101010 DPD C! FF PD C! PD C@ 3F AND DUP CB. 3F XOR OR
28              B% 10101010 DPD C! 00 PD C! PD C@ 3F AND DUP CB.        OR
29              B% 01010101 DPD C! FF PD C! PD C@ 3F AND DUP CB. 3F XOR OR
30              B% 01010101 DPD C! 00 PD C! PD C@ 3F AND DUP CB.        OR DUP CH.
31  CR CH. COLD |>
```

Testing Forth

Jörg Völker

Der bekannte ANS–Forth–Test hat als Ziel, die Übereinstimmung eines Forth–Systems mit dem ANS–Forth–Standard nachzuweisen. Das Konzept eignet sich aber auch hervorragend, um uns beim Portieren oder Neu–Implementieren eines Forth–Systems zu unterstützen.

Ich versuche gerade, mein hauseigenes 32–Bit–Fancy–Forth vom STM8 auf MC9S08–Mikrocontroller zu portieren. Auch wenn — oder gerade weil? — sich diese beiden Controller von Struktur und Befehlssatz sehr ähnlich sind, bleiben Fehler bei der Portierung natürlich nicht aus. Solange der Forth–Kern selbst noch nicht läuft, ist die Fehlersuche durch den gefädelten Code nicht ganz einfach. Meldet sich das System erstmal mit `OK`, ist das Schlimmste überstanden und man kann interaktiv weiter testen. Da bietet es sich dann an, die Interpreter–Eigenschaften zu nutzen und mit Test–Scripts zu arbeiten, denn die Tests müssen wahrscheinlich viele Male wiederholt werden, bis dann wirklich alles läuft.

Der ANS–Forth–Test von John Hayes (einfach mal „googlen“) zeigt, wie es gehen kann. Hier wird nach einer Operation der Ist–Zustand auf dem Stack mit einem vorgegebenen Soll–Zustand verglichen. Die Sache hat nur einen Haken: Die Test–Suite selbst ist in Forth geschrieben und setzt ein weitgehend funktionsfähiges Forth voraus, unter anderem mit `DO...LOOP`, `IF...ELSE...THEN`, `S"`, `TYPE`, `CONSTANT` usw. Für einen Test auf Konformität ist das in Ordnung, für die Fehlersuche in einem frisch aufgesetzten Forth aber keine große Hilfe, es wird viel zu viel vorausgesetzt. Was tun? Assembler!

Die Vorlage basiert auf gerade einmal drei Forth–Worten:

`T{` leitet eine Test–Sequenz ein. In der Forth Variante ist das tatsächlich nur „syntactic sugar“.

`->` kopiert den aktuellen Ist–Stack–Inhalt in ein Array.

`}T` vergleicht den Soll–Stack–Inhalt nach dem `->` mit dem im Array gespeichertem Ist–Zustand vor dem `->`. Anschließend wird der Stack zurückgesetzt, was in Forth reichlich mühsam ist (siehe Listing 1).

Dazu wird noch überprüft, ob die Anzahl der Stack–Einträge vor und nach dem `->` übereinstimmen.

Tatsächlich ist es verblüffend einfach, diese drei Worte direkt in Assembler umzusetzen. Den Stack zu kopieren, ist dabei auch nicht notwendig. Der interne Ablauf ist etwas anders, die Funktionalität aber identisch:

`T{` speichert den aktuellen Stackpointer in eine Variable `ForthTemp`.

`->` speichert den aktuellen Stackpointer in eine Variable `ForthTemp2`.

`}T` vergleicht den Ist–Stack–Inhalt zwischen `ForthTemp` und `ForthTemp2` mit den Sollwerten zwischen `ForthTemp2` und dem aktuellem Stackpointer.

Die Assembler–Listings sind recht *speziell*, eben angepasst auf meine Systeme, aber als Anhaltspunkt vielleicht doch geeignet. Das Umsetzen auf andere Prozessortypen und Forth–Implementierungen sollte nicht so schwierig sein. Dann braucht man nur noch die sehr umfangreichen Tests von John Hayes und seinen Kollegen, vielleicht ein bisschen anders sortiert und angepasst, und kann quasi auf Knopfdruck sein System auf Herz und Nieren durchchecken.

Fazit

Wer Forth portiert oder so wie ich ständig an seinem Forth–Kernen herumschraubt, braucht sowas!

Listing 1

zeigt die originale Forth–Quelle.

```
1   \ Date: Mon, 27 Nov 95 13:10:09 PST
2
3   \ (C) 1995 JOHNS HOPKINS UNIVERSITY / APPLIED PHYSICS LABORATORY
4   \ MAY BE DISTRIBUTED FREELY AS LONG AS THIS COPYRIGHT NOTICE REMAINS.
5   \ VERSION 1.1
6
7   \ 22/1/09 The words { and } have been changed to T{ and }T respectively to
8   \ agree with the Forth 200X file ttester.fs. This avoids clashes with
9   \ locals using { ... } and the FSL use of }
10
11  HEX
12
13  \ SET THE FOLLOWING FLAG TO TRUE FOR MORE VERBOSE OUTPUT; THIS MAY
14  \ ALLOW YOU TO TELL WHICH TEST CAUSED YOUR SYSTEM TO HANG.
15  VARIABLE VERBOSE
16          FALSE VERBOSE !
17  \       TRUE VERBOSE !
18
```

```
19  : EMPTY-STACK   \ ( ... -- ) EMPTY STACK: HANDLES UNDERFLOWED STACK TOO.
20     DEPTH ?DUP IF DUP 0< IF NEGATE 0 DO 0 LOOP ELSE 0 DO DROP LOOP THEN THEN ;
21
22  : ERROR         \ ( C-ADDR U -- ) DISPLAY AN ERROR MESSAGE FOLLOWED BY
23                  \ THE LINE THAT HAD THE ERROR.
24     TYPE SOURCE TYPE CR                  \ DISPLAY LINE CORRESPONDING TO ERROR
25     EMPTY-STACK                          \ THROW AWAY EVERY THING ELSE
26  \       QUIT  \ *** Uncomment this line to QUIT on an error
27  ;
28
29  VARIABLE ACTUAL-DEPTH                   \ STACK RECORD
30  CREATE ACTUAL-RESULTS 20 CELLS ALLOT
31
32  : T{            \ ( -- ) SYNTACTIC SUGAR.
33     ;
34
35  : ->            \ ( ... -- ) RECORD DEPTH AND CONTENT OF STACK.
36     DEPTH DUP ACTUAL-DEPTH !             \ RECORD DEPTH
37     ?DUP IF                              \ IF THERE IS SOMETHING ON STACK
38        0 DO ACTUAL-RESULTS I CELLS + ! LOOP \ SAVE THEM
39     THEN ;
40
41  : }T            \ ( ... -- ) COMPARE STACK (EXPECTED) CONTENTS WITH SAVED
42        \ (ACTUAL) CONTENTS.
43     DEPTH ACTUAL-DEPTH @ = IF            \ IF DEPTHS MATCH
44        DEPTH ?DUP IF                     \ IF THERE IS SOMETHING ON THE STACK
45           0  DO                          \ FOR EACH STACK ITEM
46                 ACTUAL-RESULTS I CELLS + @      \ COMPARE ACTUAL WITH EXPECTED
47                 <> IF S" INCORRECT RESULT: " ERROR LEAVE THEN
48           LOOP
49        THEN
50     ELSE                                 \ DEPTH MISMATCH
51        S" WRONG NUMBER OF RESULTS: " ERROR
52     THEN ;
```

Listing 2

zeigt eine Umsetzung in S08 Assembler.

```
1   ;       Fancy S08
2   ;       ####  ###   ###   #####  #  #
3   ;       #    #   #  #  #    #    #  #
4   ;       ###  #   #  ###     #    ####
5   ;       #    #   #  #  #    #    #  #
6   ;       #     ###   #  #    #    #  #
7   ;
8   ;       Test.inc (c) tematik GmbH Voe 11/2022
9   ;
10  ;       startTestToken  equ     (*-dictionary)/2
11  ;                       dc.w    startTest       ; T{ ( -- )
12  ;       testArrowToken  equ     (*-dictionary)/2
13  ;                       dc.w    testArrow       ; -> ( -- )
14  ;       evalTestToken   equ     (*-dictionary)/2
15  ;                       dc.w    evalTest        ; }T ( -- )
16  ;
17  ;       T{ doSomething -> expectedResult }T
18
19  startTest       tsx             ; TOS address at start
20                  sthx    forthTemp
21                  next
22
23  testArrow       tsx             ; TOS after doSomething
24                  sthx    forthTemp2
25                  next
26
27  evalTest        tsx                     ; compares stack
28                  pshhx                   ; right pointer
29                  ldhx    forthTemp2
30                  pshhx                   ; left pointer
31  eLoop           ldhx    1,sp
32                  cphx    forthTemp       ; done?
33                  beq     eTest
34                  bhs     eFail           ; stack overflow
35                  lda     ,x              ; left stack byte
36                  aix     #1
```

```
37                  sthx    1,sp            ; left pointer +1
38                  ldhx    3,sp            ; right pointer
39                  cmp     ,x
40                  bne     eFail
41                  aix     #1
42                  sthx    3,sp            ; right pointer
43                  bra     eLoop
44
45  eTest           ldhx    3,sp            ; right pointer
46                  cphx    forthTemp2
47                  bne     eFail
48
49  eExit           ldhx    forthTemp       ; everything's fine
50                  txs                     ; no pops needed
51                  next
52
53  eMessage        dc.b    '*Fail* ',0
54
55  eFail           ldhx    #eMessage
56  ef1             lda     ,x
57                  beq     eExit
58                  jsr     codeEmit
59                  incx
60                  bra     ef1
61
```

Listing 3

zeigt eine Variante für den STM8.

```
1   ;       Fancy STM8
2   ;       ####  ###   ###   #####  #  #
3   ;       #    #   #  #  #    #    #  #
4   ;       ###  #   #  ###     #    ####
5   ;       #    #   #  #  #    #    #  #
6   ;       #     ###   #  #    #    #  #
7   ;
8   ;       Test.inc (c) tematik GmbH Voe 11/2022
9   ;
10  ;       startTestToken  equ     {{*-dictionary}/2
11  ;                       dc.w    startTest       ; T{ ( -- )
12  ;       testArrowToken  equ     {{*-dictionary}/2
13  ;                       dc.w    testArrow       ; -> ( -- )
14  ;       evalTestToken   equ     {{*-dictionary}/2
15  ;                       dc.w    evalTest        ; }T ( -- )
16  ;
17  ;       T{ doSomething -> expectedResult }T
18
19  startTest       ldw     x,sp            ; TOS address at start
20                  ldw     forthTemp,x
21                  nextY
22
23  testArrow       ldw     x,sp            ; TOS after doSomething
24                  ldw     forthTemp2,x
25                  nextY
26
27  evalTest        ldw     x,sp            ; compares stack
28                  pushw   x               ; right pointer
29                  ldw     x,forthTemp2
30                  pushw   x               ; left pointer
31  eLoop           ldw     x,(1,sp)        ; left pointer
32                  cpw     x,forthTemp
33                  jreq    eTest
34                  jrugt   eFail           ; stack overflow
35                  ld      a,(1,x)         ; left stack byte
36                  incw    x
37                  ldw     (1,sp),x        ; left pointer
38                  ldw     x,(3,sp)        ; right pointer
39                  cp      a,(1,x)         ; right stack byte
40                  jrne    eFail
41                  incw    x
42                  ldw     (3,sp),x        ; right pointer
43                  jra     eLoop
44
45  eTest           ldw     x,(3,sp)        ; right pointer
```

```
46                  cpw     x,forthTemp2
47                  jrne    eFail
48
49  eExit           ldw     x,forthTemp     ; everything's fine
50                  ldw     sp,x            ; no pops needed
51                  nextY
52
53  eMessage        dc.b    "*Fail*",0
54
55  eFail           ldw     x,#eMessage
56  ef1             ld      a,(0,x)
57                  jreq    eExit
58                  call    codeEmit        ; low level emit
59                  incw    x
60                  jra     ef1
```

Listing 4

zeigt ein Beispiel, wie Tests definiert werden.

```
1   cr cr .( TESTING STACK OPS: 2DROP 2DUP 2SWAP ?DUP DEPTH DROP DUP OVER ROT SWAP ) cr
2
3   1 . T{ 1 2 2DROP -> }T
4   2 . T{ 12345 23456 2DUP -> 12345 23456 12345 23456 }T
5   3 . T{ 1 2 3 4 2OVER -> 1 2 3 4 1 2 }T
6   4 . T{ 12345 23456 54321 65432 2SWAP -> 54321 65432 12345 23456 }T
7   5 . T{ 0 ?DUP -> 0 }T
8   6 . T{ 1 ?DUP -> 1 1 }T
9   7 . T{ -1 ?DUP -> -1 -1 }T
10  8 . T{ DEPTH -> 0 }T
11  9 . T{ 0 DEPTH -> 0 1 }T
12  10 . T{ 0 1 DEPTH -> 0 1 2 }T
13  11 . T{ 0 DROP -> }T
14  12 . T{ 1 2 DROP -> 1 }T
15  13 . T{ 12345 DUP -> 12345 12345 }T
16  14 . T{ 12345 23456 OVER -> 12345 23456 12345 }T
17  15 . T{ 12345 23456 54321 ROT -> 23456 54321 12345 }T
18  16 . T{ 12345 23456 SWAP -> 23456 12345 }T
```

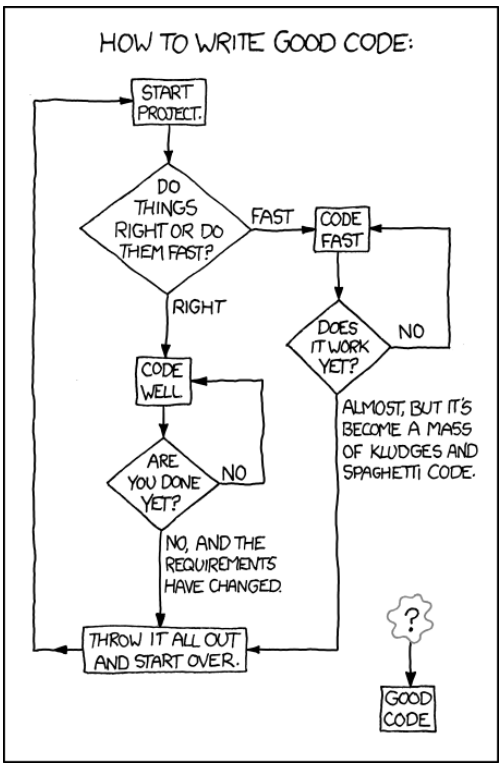


Abbildung 1: Permanent link to this comic: `https://xkcd.com/844/`

Chess–Board auf dem WikiReader

Klaus Kohl–Schöpe

Da ich mich für Schach interessiere und einige Bücher mit in den Urlaub nehmen wollte, suchte ich nach einer Möglichkeit, die Züge aus den Büchern nachzuvollziehen. Aber da ich dann meist den ganzen Tag in der Sonne wäre, war das Handy keine Option, sondern es sollte ein Low–Power–LCD–Gerät sein. Glücklicherweise ist mir eingefallen, dass ich noch ein paar WikiReader habe, die genau dem Wunsch entsprechen und auch in Forth programmierbar sind.

Der WikiReader

ist ein Projekt aus dem Jahr 2009, um Inhalte der Wikipedia offline zu lesen (Abb. 1). Es basiert auf dem Epson S1C33E07–Mikrocontroller mit 64 KB Flash, 32 MB SDRAM und einer microSD–Karte. Das Gerät ist mit 10x10x2 cm^3 klein, hat ein 240x208–Pixel–Touch–Display und drei Tasten. Zwei AAA–Batterien sollten für 90 h Betrieb (1300 h lesen) ausreichen. Neben Anzeige der Wikipedia oder Büchern des Gutenberg–Projektes kann es auch Forth–Programme ausführen.

Abbildung 1: Der WikiReader

Leider wurde 2014 das Projekt ohne Begründung eingestellt, aber ich konnte mir dank Dirk Brühl noch einige Exemplare bei amazon.com sichern. Auch wird es immer schwieriger, neben dem Sourcecode auf GitHub weitere Unterlagen dazu zu finden. Glücklicherweise gibt es einen Forth–Simulator für Windows, den ich für die Entwicklung nutzen konnte.

Das Forth

Christopher Hall von Openmoko hat ein fast ANSI–konformes Forth mit einem 8x13–Zeichensatz implementiert, welches beim Einschalten mit gedrückter Random–Taste ein Menü mit Auswahl von Programmen aus dem File `bootmenu.4th` auf der SD–Karte anzeigt (Abb. 2).

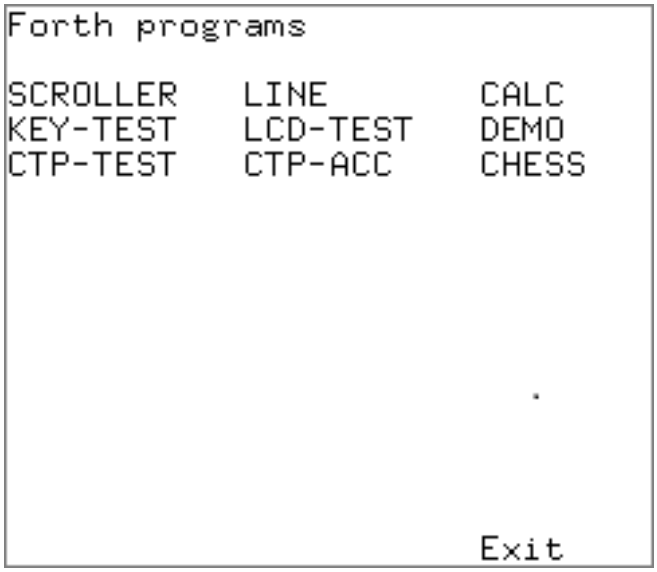


Abbildung 2: Forth–Menü

Neben diversen Testprogrammen und einem Rechner ist hier ist auch schon mein Programm `CHESS` gelistet, welches ich vorstellen will. Es sollte vorerst nicht vollständig Schach spielen können, sondern nur Züge anzeigen oder wieder zurücknehmen können. Wie es aktuell aussieht, zeigt Abb. 3.

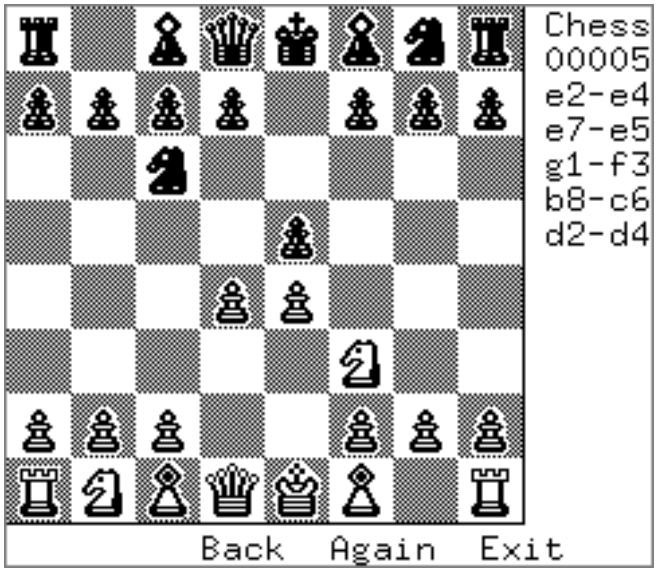


Abbildung 3: Schachbrett

Damit kommen wir auch schon zu den verschiedenen Aufgaben, die mit einem solchen Programm verbunden sind:

- Erstellung des Zeichensatzes
- Zugeingabe über Touch
- Zugrücknahme

Der Zeicheneditor

Da der Bildschirm 208 Pixel hoch ist und auch noch die Tastenbelegung und die Züge angezeigt werden sollten, nutze ich 24x24 Pixel pro Feld. Mit 8 unterschiedlichen Figuren in zwei Farben auf weißen und schwarzen Feldern braucht man 32 (+2 für leere Felder) Bilder. Ich bin nicht besonders stolz, was ich hier gezeichnet habe, aber das Programm war in einer Woche realisiert und die Figuren unterscheidbar.

Da ich damals (während KKForth–Zeiten ab 1990) bei meinem Screeneditor auch unterschiedliche Zeilenanzahlen und –längen eingeben konnte, habe ich das C64–Format 40x25 für den Zeicheneditor gewählt. Damit bleibt neben den 24 Pixeln auch Platz für den Hex–Code (Abb. 4). Ich muss nur die 24 Zeichen pro Zeile lesen und in 3 Bytes wieder speichern.

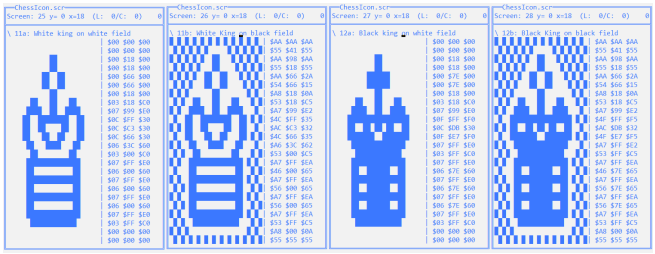


Abbildung 4: Icon–Editor

Die Routine zur Berechnung des Hex–Codes (`hexicons`) und deren Speicherung in das Programmfile (`saveicons <name>`) sind auch im File `ChessIcon.scr` , aber mit üblicher 16x64–Formatierung (siehe Listing 1). Das Ergebnis habe ich dann als Array in mein Programm `Chess.4th` übernommen.

Der Simulator

Schon meine Arbeiten mit FPGAs zeigten mir, dass eine Entwicklung auf der Zielhardware schwierig und oft langsam ist. Vor allem, wenn man Programme über die serielle Schnittstelle übertragen oder sogar als Image auf eine SD–Karte brennen muss. Deshalb war ich froh, dass es für dieses WikiReader–Forth einen Windows–Simulator gibt. Es gibt zwar kleine Unterschiede (z. B. das fehlende `cmove`), die man aber leicht korrigieren kann.

Im Endeffekt muss man nur drei Files im Simulatorverzeichnis bzw. auf der SD–Karte editieren oder hinzufügen. Als Erstes muss man das File `bootmenu.4th` um den Eintrag

```
s" CHESS.4MU" add-item
```

im Befehl `read-items` ergänzen. Das hier angegebene `CHESS.4MU` ist sehr kurz:

```
include chess.4th
chess
```

Es läd dann das eigentliche Programm `CHESS.4TH`, was im WikiReader ein paar Sekunden dauern kann.

Test im WikiReader

Im WikiReader kann man natürlich auch interaktiv mit dem Forth arbeiten, aber dazu benötigt man eine serielle Schnittstelle, die man mit etwas Geschick an dem Batteriefach herausführen kann [4]. Man muss sowieso das Gehäuse öffnen, um an die SD–Karte zu kommen, um auf dieser die drei oben genannten Files zu modifizieren bzw. dort abzulegen.

Das Programm

Die oben erstellten Icons sind der erste Teil des Programmes, wobei immer die 24x24 Pixel in 9 Bytes mal 8 Zeilen (72 Bytes) zusammengefasst werden.

Natürlich wollte ich die Züge speichern, um sie bei Bedarf auch zurückzunehmen. Deshalb merke ich mir bei jedem Zug die Ausgangsposition, die gezogene Figur, die Endposition und die ersetzte Figur.

Als Nächstes gibt es Arrays mit der Startaufstellung und das Array mit der aktuellen Aufstellung, welche die Routine `drawboard` auf das LCD zeichnet. Als Ergänzung werden auch noch die letzten 10 Züge rechts neben dem Board aufgelistet.

Beim Zeichnen ist zu beachten, dass zwar auch ein Bit pro Pixel verwendet wird, aber bei einer Zeile mit 240 Pixeln (30 Byte) zwei zusätzliche Bytes reserviert sind, welche nicht angezeigt werden. Natürlich muss man auch aufpassen, was bei der Nummerierung des Feldes auf dem Schachbrett und dann im Display oben bzw. unten ist. Teilweise hatte ich unerklärliche Effekte, weil Pixel nicht richtig dargestellt oder überschrieben wurden.

Bei den Experimenten mit Touch zeigte sich, dass eine Abfrage sinnvoll ist, ob das gewählte Feld auch mit der Figur der richtigen Farbe besetzt ist. Es wird dann das Feld markiert und darauf gewartet, bis das Ziel gewählt wird. Danach wird geprüft, ob das Zielfeld nicht mit der eigenen Farbe belegt ist, was ja nicht erlaubt ist. Ist das ok (weitere mögliche Abfragen im nächsten Kapitel), wird der Zug gespeichert und die Anzeige aktualisiert.

Damit fehlt nur noch der Hauptbefehl `chess` . Dieser übernimmt die Startaufstellung in die aktuelle Konfiguration, setzt den Zügezähler zurück und wartet auf die Eingabe. Möglich sind die Tasten für „Zurück“ bzw. „Vorwärts“ oder „Programm beenden“. Über das Touch–Display werden die Züge eingegeben, wobei auch da zuerst geprüft wird, ob ein Schachfeld angeklickt wurde. Bei der ersten Berührung wird geprüft, ob die erlaubte Figur gewählt wurde und bei der zweiten Berührung dann (leider noch unvollständig) die erlaubten Ziele.

Notwendige Verbesserungen

Leider habe ich dann das Projekt nicht mehr weiterentwickelt, obwohl noch einige Punkte notwendig wären:

- Vollständige Prüfung, ob ein Zug erlaubt ist
- „Rochade“ und „En passant“
- Bauernumwandlung
- Editor für Aufstellung

Auch ein komplettes Schachprogramm mit Eröffnungsbibliothek wäre hier möglich, aber dies werde ich vermutlich mit einer deutlich schnelleren Hardware realisieren.

Ich denke, dieses Beispiel zeigt, dass man manchmal auch alte Hardware für eine kleine Applikation heranziehen kann, wenn man spezielle Anforderungen hat. Viel Spaß damit.

Links

[1] WikiReader: `https://de.wikipedia.org/wiki/WikiReader`
[2] Offizielle WikiReader–Sourcen auf GitHub: `https://github.com/wikireader/wikireader`
[3] WikiReader–FORTH–Simulator: `http://createuniverses.blogspot.com/2011/03/wikireader-forth-simulator.html`
[4] WikiReader–Serial–Port–Hack: `http://toddbot.blogspot.com/2010/05/wikireader-serial-port-hack-pictures.html`
[5] KK–Quellcode zum Heft: `https://wiki.forth-ev.de/doku.php/vd-archiv`

Listing 1: Sourcen für den IconEditor

Die Sourcen sind für *mcForth*.

Besonderheit: `Create` und `Createp` müssen mit `End-Create` abgeschlossen werden.

```
1   Screen 1:
2   \ Loadscreen with Converter
3     : readblock    ( n -- addr ) \ read block to here
4      #1024 um* 0 fid @ fpos! here dup >x #1024 fid @ fread drop ;
5     : writeblock    ( n -- ) \ write block from here
6      #1024 um* 0 fid @ fpos! here     >x #1024 fid @ fwrite ;
7     : hexicons    ( -- ) \ Convert icons to hex code
8       base push  hex
9       capacity 1 -  3 ?DO                \ for all blocks
10        i readblock  #40 +  #24 FOR       ( -- addr ) \ all lines
11          dup #26 +  3 FOR               ( -- addr addr+26 )
12            0 swap  8 FOR                ( -- addr 0 addr+26 )
13              >r >r  dup c@ #32 - IF #219 over c!  1 ELSE 0 THEN
14                 r> 2* +  swap 1+ swap r> ( -- addr+1 n addr+26 )
15            NEXT >r 0 <# # # [char] $ hold #> r@ swap cmove r> 4 +
16          NEXT drop #16 +                ( -- addr+24+16 )
17      NEXT drop  i writeblock  LOOP ;
18
19  Screen 2:
20  \ Save Icons
21    Createp crlf  $0d    c,p  $0a    c,p          End-Create
22    Createp "c,"  char c c,p  char , c,p  $20 c,p End-Create
23    : saveicons   ( file ; -- )
24      parse-name  $>h count  fcreate           \ new file
25      capacity 1 -  3 ?DO                      \ for all blocks
26    i readblock over >r  dup #40 -trailing >r >x r> r> fwrite
27        #66 +   over >r  crlf p>x 2 r> fwrite
28        #8 FOR
29          3 FOR  3 FOR
30                over >r  dup    >x 4 r> fwrite  4 +
31                over >r  "c,"  p>x 3 r> fwrite
32            NEXT  #28 +
33          NEXT  over >r  crlf  p>x 2 r> fwrite
34        NEXT drop
35      LOOP fclose ;                   \ close file
36
```

Listing 2: Sourcen für das Chess–Board im WikiReader

Listing gekürzt um ca. drei Seiten Komma–Code der Icons. Die komplette Quelle findest du im Forth–e.V–Wiki [5].

```
1   \ ------------------------------------------------------------
2   \ Chess for Wikireader
3   \ Author: Klaus Kohl-Schoepe
4   \ ------------------------------------------------------------
```

```
\ Description:
\  Display: Start at lcd-vram with 32 byte width (30 visible)
\  Board: 192x192 Pixel (24*24 fields) with text on the right
\ -------------------------------------------------------------

base @  decimal

\ -------------------------------------------------------------
\ Chess Icons
\ -------------------------------------------------------------
\ Figures: *2 for white field and *2+1 for black field
\  0=empty; 1/2=white/black pawn; 3/4=bishop
\  4/5=knight; 6/7=rook, 8/9=bishop, 10/11=king

24 constant fieldsize

create icons

...

\ 1a: White pawn on white field
$00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c,
$00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c,
$00 c, $18 c, $00 c, $00 c, $3C c, $00 c, $00 c, $66 c, $00 c,
$00 c, $66 c, $00 c, $00 c, $3C c, $00 c, $00 c, $FF c, $00 c,
$01 c, $81 c, $80 c, $00 c, $FF c, $00 c, $00 c, $66 c, $00 c,
$00 c, $66 c, $00 c, $00 c, $FF c, $00 c, $01 c, $81 c, $80 c,
$03 c, $FF c, $C0 c, $03 c, $00 c, $C0 c, $03 c, $FF c, $C0 c,
$01 c, $FF c, $80 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c,
\ 1b: White pawn on black field
$AA c, $AA c, $AA c, $55 c, $55 c, $55 c, $AA c, $AA c, $AA c,
$55 c, $55 c, $55 c, $AA c, $AA c, $AA c, $55 c, $41 c, $55 c,
$AA c, $98 c, $AA c, $55 c, $3C c, $55 c, $AA c, $66 c, $2A c,
$54 c, $66 c, $55 c, $AA c, $3C c, $2A c, $54 c, $FF c, $15 c,
$A9 c, $81 c, $AA c, $54 c, $FF c, $15 c, $AA c, $66 c, $2A c,
$54 c, $66 c, $15 c, $A9 c, $FF c, $AA c, $51 c, $81 c, $95 c,
$AB c, $FF c, $CA c, $53 c, $00 c, $D5 c, $AB c, $FF c, $CA c,
$51 c, $FF c, $95 c, $A8 c, $00 c, $2A c, $55 c, $55 c, $55 c,

...

\ -------------------------------------------------------------
\ Fields and variables
\ -------------------------------------------------------------
\ Save chessmoves in an array
200 constant #maxmoves \ actual maximum 200 moves
variable chessmoves    \ actual move position
variable maxmoves      \ maximum moves logged
create movelist ( old pos; figure; new pos; replaced figur )
   #maxmoves 4 * allot
variable fromfield \ flag for moving: -1: new; 0-63: first position

\ Chess start board
create startboard
   7 c,  5 c,  3 c,  9 c, 11 c,  3 c,  5 c,  7 c,
   1 c,  1 c,  1 c,  1 c,  1 c,  1 c,  1 c,  1 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   2 c,  2 c,  2 c,  2 c,  2 c,  2 c,  2 c,  2 c,
   8 c,  6 c,  4 c, 10 c, 12 c,  4 c,  6 c,  8 c,

\ Actual displayed board
create dispboard
   7 c,  5 c,  3 c,  9 c, 11 c,  3 c,  5 c,  7 c,
   1 c,  1 c,  1 c,  1 c,  1 c,  1 c,  1 c,  1 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,  0 c,
   2 c,  2 c,  2 c,  2 c,  2 c,  2 c,  2 c,  2 c,
   8 c,  6 c,  4 c, 10 c, 12 c,  4 c,  6 c,  8 c,

\ -------------------------------------------------------------
\ Draw board (0 = a1 left bottom is black)
```

```
\ ------------------------------------------------------------
: drawfield ( field figure -- ) \ draw one field
  2* >r  dup dup 3 rshift + 1 and  0= if r> 1 +  >r then \ Figure+color
  63 over -  3 rshift  768 *
  swap       7 and       3 *  +  lcd-vram + \ LCD address
  r> 72 *  icons  +                        \ icon Address
  24 0 do
    3 0 do dup >r c@  over c!  1 +  r> 1 + loop
    >r 29 +  r>
  loop
  drop drop ;

: pos.      ( field -- ) \ print a1..h8
  dup 7 and  $61 + lcd-emit
  3 rshift   $31 + lcd-emit ;

: drawboard ( -- )
  lcd-cls
  64 0 do i  i dispboard + c@  drawfield loop
  fieldsize 8 *  0              lcd-move-to
  fieldsize 8 *  dup            lcd-line-to
  0              fieldsize 8 *  lcd-line-to
  25  0   lcd-at-xy   s" Chess"                      lcd-type
  25  1   lcd-at-xy   chessmoves @ 0 <# # # # # # #>  lcd-type
  25  2   chessmoves @  dup  dup 13 min  -     \ last 10 moves
  ?do over over lcd-at-xy  1 +
      i 4 * movelist +  dup c@ pos.  s" -" lcd-type  2 + c@ pos.
  loop drop drop
  9   lcd-text-rows 1-  lcd-at-xy   s" Back  Again  Exit" lcd-type ;

\ ------------------------------------------------------------
\ Main program with move handling
\ ------------------------------------------------------------
: allowfrom?   ( field -- field -1 | 0 ) \ check from
  dup dispboard + c@          \ field figure
  ?dup 0= if drop 0 exit then \ no empty field
  chessmoves @  xor  1 and    \ right color ?
  if -1 else drop 0  then ;

: allowto?     ( field -- field -1 | 0 ) \ check to
  fromfield @  over  = if drop 0 exit then    \ same field not allowed
  dup dispboard + c@  ?dup 0= if -1 exit then \ empty field allowed
  chessmoves @  xor  1 and                    \ right color ?
  if drop 0 else -1 then ;

: -move   ( -- ) \ backward one move
  chessmoves @  ?dup
  if   1 -  dup chessmoves !  4 *  movelist +   \ movelist
           dup 1 + c@  over c@  dispboard +  c! \ back to old position
       2 + dup 1 + c@  swap c@  dispboard +  c! \ restore field
       drawboard
  then ;

: +move   ( -- ) \ forward one move
  chessmoves @   dup  maxmoves @  <
  if   dup  1 +  chessmoves !  4 *  movelist +  \ movelist
       0           over     c@  dispboard +  c! \ remove from old position
       dup 1 + c@  swap 2 + c@  dispboard +  c! \ set to new field
       drawboard
  else drop
  then ;

: setmove ( from to -- ) \ save move
  chessmoves @   dup  #maxmoves  <
  if   dup  1 +  dup chessmoves !  maxmoves !
           4 * movelist +    >r
       over                          r@     c! \ old position
       swap dispboard +  dup c@   dup r@ 1 + c! \ figure
       >r  0 swap c!  r>                       \ empty old positon
       over                          r@ 2 + c! \ new position
       swap dispboard +  dup c@      r> 3 + c! \ replaced figure
                  c!                           \ figure to new position
       drawboard
  else drop drop drop
  then ;
```

```
157   : newmove   ( -- ) \ with highlight of first field
158     ctp-pos ctp-flush
159     dup fieldsize 8 * 0 within  over fieldsize 8 * 0 within or
160     if drop drop exit then
161     fieldsize /  7 swap -  8 *  swap fieldsize /  +
162     fromfield @  -1  =
163     if   allowfrom? 0= if exit then   \ ignore wrong figure/field
164          dup fromfield !                \ save position
165          dup 7 and                 fieldsize *  \ x*size
166          swap  3 rshift  7 swap -  fieldsize *  \ y*size
167          over over  lcd-move-to
168          over                 over fieldsize + 1-  lcd-line-to
169          over fieldsize + 1-  over fieldsize + 1-  lcd-line-to
170          over fieldsize + 1-  over                 lcd-line-to
171                                                    lcd-line-to
172     else allowto? if fromfield @  swap  setmove   -1 fromfield ! then
173     then ;
174
175   : chess ( -- )
176     startboard dispboard 64 cmove
177     0 chessmoves !  0 maxmoves !
178     drawboard
179     button-flush   ctp-flush   -1 fromfield !
180     begin
181        wait-for-event
182        button? if button  button-flush  -1 fromfield !
183                   dup button-right  = if drop exit then
184                   dup button-left   = if -move then
185                       button-centre = if +move then
186                then
187        ctp-pos? if newmove then
188     again ;
189
190   \ ------------------------------------------------------------
191
192   base !
```

Interesting Programming Languages

Seit Februar 2020 pflegt Pradeep Gowda diese Webseite, zuletzt aktualisiert am 01.12.2022. Er hat da alles gesammelt, was ihm irgendwie interessant vorkam. So ist es subjektiv, dennoch inzwischen eine erstaunliche Zusammenstellung mit kurzen Beschreibungen der Sprachen. Und, kommt Forth darin vor? Natürlich, 2 mal.

> „Reva — a small, cross–platform Forth. Recommended by \@deech in the context of ‘read the source code of the programming language to learn how things work’“

und

> „Play – a statically typed stack language (aka Forth)“

Im Übrigen staunt man, was es da so alles gibt in dieser Fundgrube. Allerdings sieht man da auch gut, wie schnelllebig so manches der angepriesenen Produkte ist. Wenn die verlinkten Seiten gar nicht mehr existieren, oder auf dem Link seit zwei oder mehr Jahren nichts weiter passiert ist, der tollen Ankündigung nichts folgte, dann ahnt man schon die Vaporware dahinter. Forth dagegen gibt es noch heute.

Schaut mal rein in den Link. cas/mk

`https://www.btbytes.com/pl.html`

(Quelle: Irgendwo aus dem Internet ... :)

Forth–Gruppen regional

Bitte erkundigt euch bei den Veranstaltern, ob die Treffen stattfinden. Das kann je nach Pandemie–Lage variieren.

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

München **Bernd Paysan**
Tel.: (0 89) – 41 15 46 53
`bernd@net2o.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00 auf `http://public.senfcall.de/forth-muenchen`, Passwort `over+swap`.

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

Ruhrgebiet **Carsten Strotmann**
`ruhrpott-forth@strotmann.de`
Treffen alle 1–2 Monate im Unperfekthaus Essen
`http://unperfekthaus.de`.
Termine unter: `https://www.meetup.com/Essen-Forth-Meetup/`

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 microcore (uCore)
Klaus Schleisiek
Tel.: **(0 58 46) – 98 04 00 8**$_p$
`kschleisiek@freenet.de`

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

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

Termine

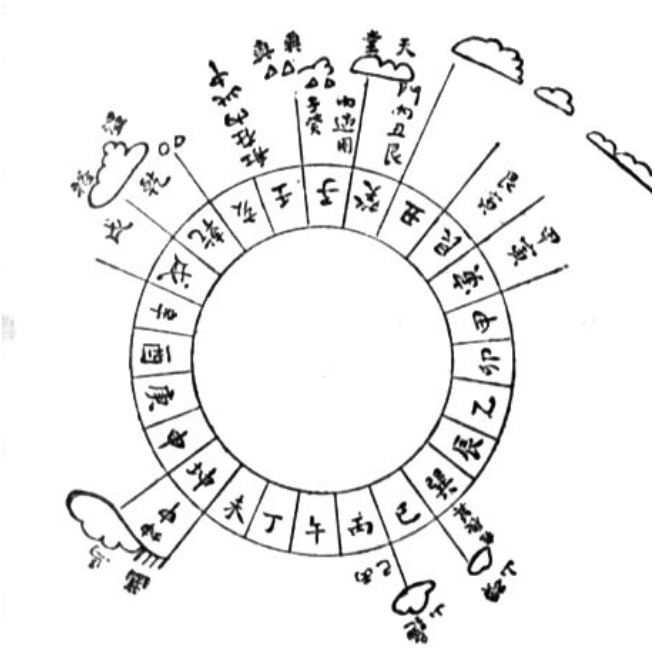
Donnerstags ab 20:00 Uhr
Forth–Chat net2o `forth@bernd` mit dem Key `keysearch kQusJ`, voller Key:
`kQusJzA;7*?t=uy@X}1GWr!+0qqp_Cn176t4(dQ*`

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 im Monat
ZOOM–Treffen der Forth2020 Facebook–Gruppe
Infos zur Teilnahme: `www.forth2020.org`

Forth–Tagung (online) 25.–26. März 2023
`https://tagung.forth-ev.de`

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



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

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

Einladung zur Forth–Tagung 2023 am 25. und 26. März 2023 (online)

Organisation: Vorstandskreis der FG

Im März 2023 wird die Forth–Gesellschaft eine Kurz–Tagung (Samstag 26.3.2023) und die Mitgliederversammlung online durchführen. Da noch nicht alle Mitglieder das Risiko einer Präsenzveranstaltung eingehen können oder wollen, wird diese Tagung online mit dem schon bewährten BBB–System stattfinden.

Im Sommer (08.06. — 11.06.2023) planen wir eine Sommertagung mit persönlichen Treffen. Ort und Details werden voraussichtlich im Heft 2023–01 unseres Forth–Magazins, der *Vierten Dimension*, an dieser Stelle bekanntgegeben.

Anmeldung

Auf `https://tagung.forth-ev.de` findet ihr weitere Informationen dazu, das aktuelle Programm sowie die Möglichkeit, euch zur Tagung elektronisch anzumelden.

Programm

Freitag 24.03.2023

- Abends: Informeller Treff online ohne besonderes Programm;
  ab 19:00 Uhr

Samstag 25.3.2023

- Vormittag: Vorträge und Workshops
- Nachmittag: Vorträge und Workshops
- Abend: Gemeinsames Abendessen (Online–Edition)

Sonntag: 26.3.2023

- Mitgliederversammlung der Forth–Gesellschaft e.V.;
  10:00 — 13:00 Uhr
- Platz für weitere Workshops;
  15:00 — 18:00 Uhr

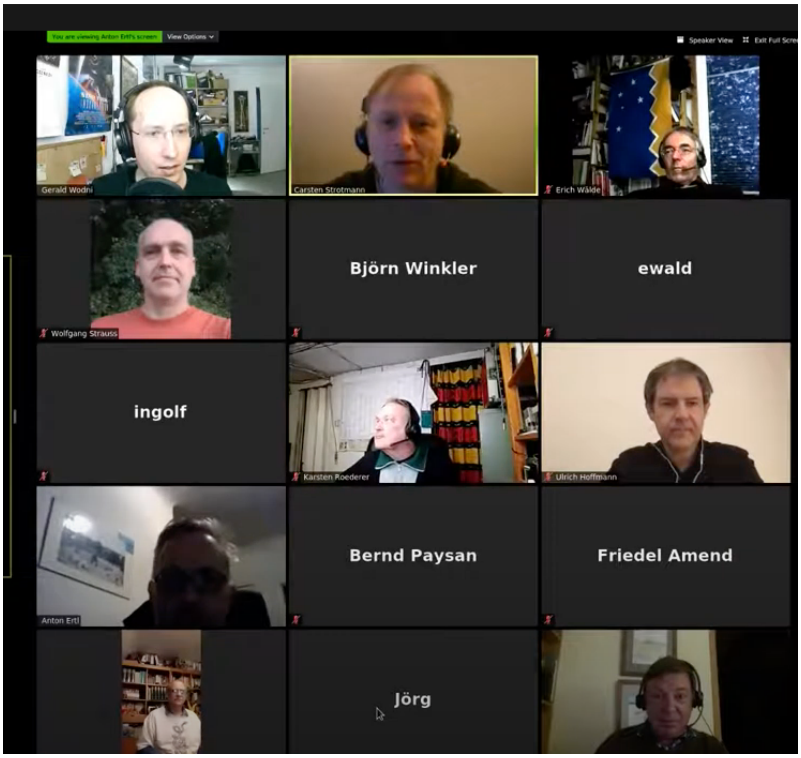


Abbildung 1: Anfang diesen Jahres sah das dann so aus wie auf diesem Bild.
Probiert schon mal eine gute Beleuchtung und einen hübschen Hintergrund für euch aus.