Vierte Dimension

2/2014

Das Forth–Magazin

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

In dieser Ausgabe:

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

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

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Liebe Leser,

Die Forth-Gemeinschaft ist lebendig wie lange nicht mehr. Es gibt viel zu berichten, viel mehr als wir Platz in der VD haben. Diese Ausgabe hat 40 Seiten, es hätten noch mehr sein können. Doch diese Ausgabe ist (im Juli 2014) "überreif" und muss zum Leser. Die Verspätung dieser VD ist meiner "Brötchenverdienst"–Arbeit im April-Juni zu verdanken.

Die Tagung im Bad–Vöslau hat viele neue Impulse gebracht, neue Arbeitskreise und Projekte wurden geboren. Bernd Paysan berichtet in dieser Ausgabe über das Open–Network–Forth–Projekt und die neue Implementierung auf dem Tiva–Connected–Launchpad.

eBooks und "Print–on–demand" sind eine Chance, Forth im Buchmarkt sichtbarer zu machen. Neue Bücher erscheinen über diesen Vertriebsweg, und auch vergriffene Bücher können so wieder verfügbar gemacht werden.

Die Forth-Gesellschaft nimmt Kontakt auf zu Regional-Treffen des CCC, dort wird viel mit Hard- und Software gebastelt. Lötkolben sind dort vorhanden, die Projekte und Herangehensweisen an Ausgaben "kompatibel". Im Bericht von der Mitgliederversammlung stehen die Termine, es dürfen aber noch mehr werden!

Auch wenn mein Name in diesem Heft unter dem Editorial steht, so ist diese Ausgabe wieder ein Gemeinschaftswerk des VD-Redaktionsteams, allen voran der Autoren, speziell Fred Behringer und Michael Kalus, die präzise und zeitnah Ideen und Korrekturen einbringen. Nur die verbleibenden Fehler dieser Ausgabe sind die meinigen.

Da wir nicht alles über Forth in dieser Ausgabe unterbringen konnten, wird es weitere VD-Ausgaben geben, auch neue Sonderhefte (ARM-CPU Sonderheft) sind geplant. Bis dahin wünsche ich viel Spaß mit dieser Ausgabe.

Carsten Strotmann

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-2014-02`

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

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

Neues Buch über Factor

Unter dem Titel "Seven More Languages in Seven Weeks" ist bei den Pragmatic Programmern ein Buch (unter anderem) über die Sprache Factor erschienen. Factor ist eine von Forth abstammende "concatenative" Sprache mit funktionalen Erweiterungen, über Factor wurde in früheren Ausgaben der VD berichtet. Neben Factor beschreibt das Buch die Programmiersprachen Lua, Elixir, Elm, Julia, MiniKanren und Idris. Ziel dieses Buches, wie auch schon dessen Vorläufer "Seven Languages in Seven Weeks", ist es, dem Leser das Erlernen einer neuen Programmiersprache (und den damit verbundenen neuen Ideen) in je einer Woche zu ermöglichen.

Das Buch ist gedruckt oder als eBook (PDF, mobi, ePub, ohne DRM) ab US$ 25 erhältlich unter `http://pragprog.com/book/7lang/seven-more-languages-in-seven-weeks`

CS

sforth - Forth und JavaScript

Bernd Amend arbeitet an einem Forth-System in JavaScript, d.h. der Forth-Code wird in JavaScript übersetzt und kann in einem Browser, aber auch auf der Kommandozeile per node.js, ausgeführt werden.

Neben den Forth-Funktionen stehen dem Programmierer auch die Möglichkeiten der JavaScript-Sprache zur Verfügung, sowie auch aller JavaScript Erweiterungen (jQuery etc).

Webseite: `https://github.com/tptb/sforth`

CS

gForth 0.7.3

Das gForth-Team hat die Version 0.7.3 des bekannten Forth-Systems veröffentlicht. Die neue Version behebt Probleme mit neueren Versionen des GNU-C-Compilers GCC, und Integration mit dem Emacs-Text-Editor funktioniert nun mit der neuen Emacs-Version 24.

Webseite: `http://www.gnu.org/software/gforth`

CS

MikeOS mit Forth-Interpreter

MikeOS ist ein kleines, simples 16Bit-Betriebssystem für Intel-x86-PCs. Es kann von Diskette, CD-ROM oder einem USB-Laufwerk gestartet werden. MikeOS ist vollständig in 80x86-Maschinensprache geschrieben.

Haupteinsatzzweck von MikeOS ist das Vermitteln von Betriebssystem-Grundlagen. Die neue Version 4.5 enthält nun zum ersten Mal einen Forth-Interpreter.

Webseite: `http://mikeos.sf.net`

CS

Kostenlose VFX-Forth-Lite-Compiler für ARM-Cortex-M und TI-MSP430 verfügbar

Die MPE-VFX-Forth-Compiler sind nun in einer "Lite"-Version für ARM-Cortex-M und TI-MSP430-Kerne kostenlos über die MPE-Webseite zum Download verfügbar. Derzeit unterstützen die Compiler das STM32F072B-Discovery-Board und den MSP430G2553. Unterstützung für weitere Boards ist in Vorbereitung.

Das Produkt beinhaltet den Cross-Compiler auf PC sowie das Forth-System auf dem Ziel-Mikrocontroller. Die IDE des Lite-Compilers und der Terminal-Emulator laufen auf einem Windows-PC. Der generierte Code wird im Flashspeicher des Mikrocontrollers abgespeichert. Über eine USB-Verbindung oder auch eine RS232-Verbindung werden Daten oder Code geändert.

Die "LITE"-Versionen der MPE-Forth-Systeme bieten viele Funktionen der Vollprodukte, sind aber in der Codegröße beschränkt.

Ein Beta-Tester der "VFX-Forth-Lite-Compiler" hat den zweiten Platz auf dem jüngsten IET-Roboter-Triathlon-Wettbewerb belegt.

Links:
`http://www.theiet.org/events/local/193512.cfm`
`http://www.mpeforth.com/xc7lite.htm`

MK

Lisa Neigut – Serial! It's what's for breakfast

Kleines 10 Minuten-Video über Serielle-Kommunikation von der !!Con: `http://www.youtube.com/embed/J1CQz8XyWoo?rel=0`
!!Con: `http://bangbangcon.com`

CS

no-forth C&V

Albert Nijhof und Willem Ouwerkerk haben zwei neue Versionen des no-forth für MSP430-Systeme veröffentlicht. Die "C" (Compact) Variante ist für kleine Boards mit 16KB Flash gedacht, in dieser Version fehlen einige Funktionen wie z.B. Vocabularies. Die "V" (Vocabularies) Version ist die "grosse" Ausgabe des no-forth mit Unterstützung für Vocabularies.

Webseite: `http://home.hccnet.nl/anij/nof/noforth.html`

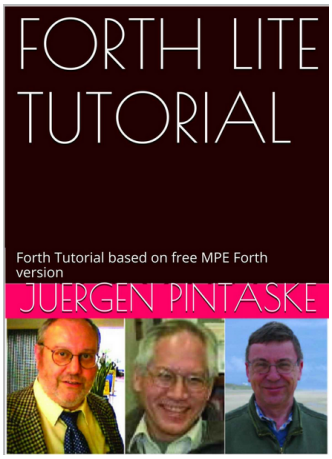
CS

Forth LITE Tutorial eBook

"FORTH LITE TUTORIAL: Forth Tutorial based on free MPE Forth version" von Juergen Pintaske

Dieses eBook ist bei Amazon im Mobi–Format erschienen. Das eBook kann auf dem Amazon-eigenen "Kindle"–eBook–Leser gelesen werden, oder per "Kindle"–App auf Apple iPad/iPhone, Android Smartphones und Tablets

sowie per Leseprogramm auf Windows- oder MacOS–X–Rechnern.

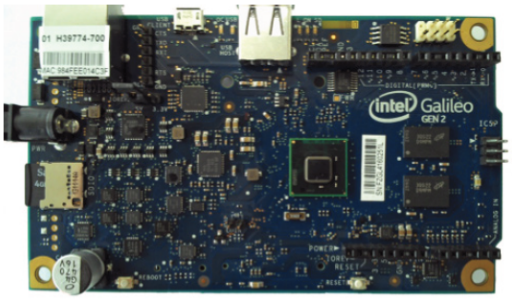


Das Buch basiert auf dem Forth–Tutorial von Leo Wong[1] und führt den Forth-Anfänger in 36 Lektionen in die Sprache ein. Entgegen dem Buchtitel ist das Buch nicht MPE–Forth–spezifisch und funktioniert mit jedem ANSI–Standard–Forth, die allermeisten Übungen und Beispiele funktionieren warscheinlich auch bei einem Forth83–System.

Das Buch ist eine leichte, interaktive Einführung in die Programmierung mit Forth. Bei einem Preis von 3.01 Euro ein Schnäppchen, bei dem man handwerkliche Schwächen bei der Aufbereitung des eBook–Formats gerne übersieht. Es ist zu hoffen, dass über den Vertriebsweg der Amazon–Webseite[2] Programmierer außerhalb der Forth–Gemeinschaft auf das Buch aufmerksam werden.

CS

Intel–Arduino–Galileo–Gen–2–Board



Intel hat für August eine neue Auflage des Arduino–Galileo–Boards angekündigt. Der Arduino–Galileo (`http://arduino.cc/en/ArduinoCertified/IntelGalileo`) ist ein Mikrokontroller-Board auf Basis des Intel–Quark–SoC (System on Chip) und ist Intels Antwort auf Raspberry Pi und andere Low–Cost–Boards. Zwar ist der Galileo weniger gut ausgerüstet als vergleichbare ARM–Boards (mit 256 MB weniger Speicher, keine Grafikhardware) bei höherem Preis (60-70 Euro), hat jedoch den Vorteil, dass der Quark–SoC auf Basis der bekannten Intel–Pentium–CPU (400 Mhz) basiert, und Entwickler x86–Wissen mitnehmen können. Betrieben wird der Galileo mit Linux, die Standard–Programmier–Umgebung ist das von anderen Arduino-Boards bekannte "sketch"–IDE. Mit ein wenig Tricks lassen sich jedoch auch andere Betriebssysteme für Intel x86–CPUs benutzen. Linux- und DOS–Forth–Systeme laufen auf dem Galileo-Board "out–of–the–box".

Das neue Galileo-Board verträgt nun Versorgungsspannungen zwischen 7 und 15 Volt, Power–over–Ethernet (per Zusatzmodul) und die serielle Schnittstelle ist nun über einen USB–TTL–Anschluss erreichbar (statt über eine spezielle Klinken–Buchse bei der ersten Version des Boards). Ein 12Bit–PWM erlaubt eine bessere Steuerung von Schrittmotoren. Einzelheiten zum Arduino–Galileo–2 finden sich im Datenblatt unter `https://communities.intel.com/docs/DOC-22795`.

Bei Heise-Developer gibt es einen Testbericht des Arduino–Galileo–SoC–Boards[3].

CS

Leserbriefe

Neues vom 434-MHz-Empfänger

In der VD 4/2011 hatte ich berichtet, wie man die Außentemperatursensoren von Technoline belauscht. Diese Lauscheinrichtung ist bei mir seit dieser Zeit lückenlos in Betrieb, die Werte des Außensensors wurden immer zuverlässig empfangen. Bis neulich. Zuerst waren nur kleinere Löcher von wenigen Stunden in den Daten. Das ist zumindest schon mal sehr ärgerlich. Ich habe parallel eine zweite Empfänger-Platine bestückt, um ein Problem mit der eingesetzten Elektronik auszuschließen. Geholfen hat das aber nichts. Im Gegenteil, die Ausfälle vergrößerten sich zu mehren bis vielen Stunden, und schließlich hatte ich in einer Woche nur noch 2 Messwerte. Das war mir dann doch zu wenig. Eine kurze Suche nach Störquellen (vermittelst DVB-T-Stick und gnuradio) war nicht erfolgreich. Dann fiel mir noch etwas ganz Triviales ein: Polarisation. Gemeinerweise kann man ja Funkwellen im Bereich 434 MHz nicht mit den Augen sehen, geschweige denn, wie sie polarisiert sind.

In einem verzweifelten "wird–ja–schon–nichts–helfen"–Versuch habe ich dann die primitive Drahtantenne einfach in die Horizontale umgebogen. Der Sender steht ungefähr senkrecht zu der Richtung, in die die Antenne zeigt. Und oh Wunder, seither sehe ich wieder einwandfrei alle Daten. Wer hätt's gedacht. Der Störer, falls wirklich vorhanden (ich tippe auf ein Babyphon), ist immer noch unbekannt, aber das ist netterweise nicht mehr wichtig.

Erich Wälde

[1] `http://www.murphywong.net/hello/simple.htm`
[2] `http://www.amazon.de/FORTH-LITE-TUTORIAL-Tutorial-version-ebook/dp/B00L0F32Q8`
[3] `http://www.heise.de/developer/artikel/Intels-Einplatinen-Computer-Galileo-unter-der-Lupe-2252539.html`

ebook vs. print on demand

Kindle Für die Texte, die Jürgen Pintaske [4] macht, kann ebook ok sein. Ich habe mir Kindle damals näher angesehen:

- Das Format (html) eignet sich nicht für Artikel mit Grafiken weil man als Autor kaum Kontrolle hat, wie das Seitenformat beim Leser aussehen wird.
- Die Seitengröße (ca. A5) des Kindle ist für Text, der auf Illustrationen verweist, zu klein.
- Die Navigation (Blättern) ist bei derzeitigen Geräten unzureichend. Man kann zwar via html-Hyperlink auf Literaturstelle am Buchende springen, aber simpler Rücksprung wie in Webbrowser überfordert das System schon.
- Es ist zwar ein elektronisches Format, aber man kann source von Programmen nicht simpel als Textfiles exportieren.
- Kopierschutz ist bei ebooks ein Problem. Amazon hat zwar DRM, aber eher lustlos, und wohl nur mittelpächtig gelöst.
- ebooks sind unschlagbar billig. Aber bei technischen Texten mit absehbar begrenzter Anzahl Lesern hilft das dem Autor wenig.

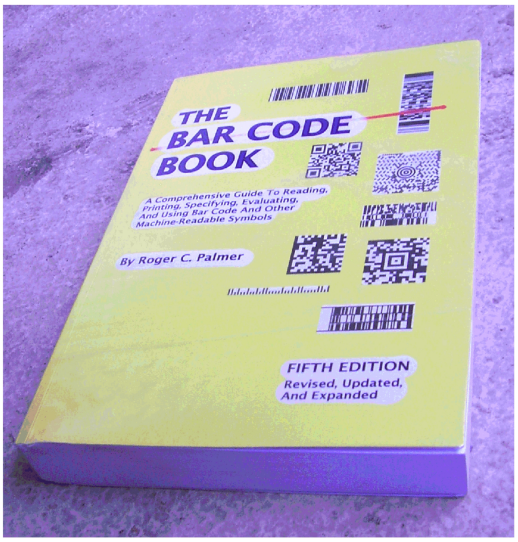
Die Marktdurchdringung für technische Texte ist deshalb bisher und absehbar weiterhin dürftig. Man ist auf den Leserkreis beschränkt der auf Kindle oder vergleichbaren Geräten wie iPad Bücher liest. Das sind wohl 10% der potentiellen Käufer.

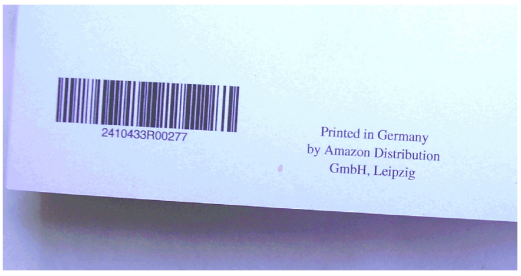
Amazon Für mich dürfte print on demand bei Amazon geeigneter sein. Es sind softcover Paperbacks in verschiedenen Formaten von etwas größer als deutsches Taschenbuch bis knapp unter A4 und Seitenzahl 24 ... 600 verfügbar. Es gäbe sie auch innen farbig, aber für Fachbücher ist eher schwarz-weiß relevant. Typisches Beispiel ist s/w, 17,78 x 25,4cm, 460 Seiten. Hatte etwa 2..3 Tage längere Lieferzeit als normales Buch ab Lager von Amazon. Liefern muss der Autor zwei pdfs: Cover und Buchblock, passend zu den Specs von Amazon. Nach technischer Prüfung und Freigabe durch Amazon kann man sich einige Exemplare produziert in Deutschland zu normalem Preis bestellen. Kiste mit kostenlosen Mustern gibts nicht. Was Amazon für Autoren dazu anbietet, kommt derzeit aus USA und wird von Lieferzeit, Transportkosten, Zoll eher nicht attraktiv sein. Dauer der Freigabe durch Amazon variiert, scheint aber unter einer Woche zu liegen, wenn man keine Fehler im pdf hat. Es gibt keine Festkosten, die Autor an Amazon zu zahlen hätte. Wie man seine eigene Marge dazu rechnet, kann man sich anhand der Vorgaben von Amazon selber ausdenken. Man kommt dann für ein Buch von 100 ... 200 Seiten s/w auf 15 EUR für den Kunden, wenn man 5 EUR selber einstreichen will. Da die Kosten nur von der Seitenzahl aber nicht von der Blattgröße abhängig sind, kann man auf dem Weg theoretisch noch etwas optimieren. Amazon ist wohl preislich deutlich günstiger als die anderen Anbieter. Die aber oft technisch weniger beschränkt sind.

An der Druckqualität wurde manchmal rumgemäkelt. So seien die Cover manchmal leicht wellig beim Kunden angekommen, weil frisch und feucht aus der Maschine sofort verschickt wurde. Auch die Schnittbreite der Ränder kann wohl geringfügig variieren. Alles im Rahmen eines üblichen Paperbacks, finde ich.

Für die ISBN gibt es Varianten, die Amazon-Nummer ist kostenlos und reicht für den Verkauf über Amazon. Man verzichtet dabei auf den Vertrieb durch den Buchhandel. Belegexemplare für die Deutsche Nationalbibliothek entfallen damit auch. Werbung fürs Buch macht Amazon nicht, das ist Job des Autors. Aber es ist voll ins Amazon-System eingebunden. Man hat damit die üblichen Funktionen wie "Blick ins Buch". MfG JRD

Amazon Book-On-Demand (Createspace):
`https://www.createspace.com/Products/Book/`





Rafael Deliano

[4] Als ebooks sind jetzt erschienen das polyFORTH-Handbuch, das Forth Lite Tutorial, und “Charles Moore” in der Kindle Edition. (Siehe: www.forth-ev.de).

Bootmanager und FAT-Reparatur: Zwölfter Fort(h)schritt (Zerschossene Partitions-Tabelle)

Fred Behringer

Ohne Partitions-Tabelle ist man (nicht nur bei 200-GB-Festplatten mit schön verteilten Betriebssystemen) aufgeworfen. Es gibt eine Reihe von Hilfsmitteln, mit denen man sich eine unbrauchbar gewordene Partitions-Tabelle wiederkonstruieren kann. Das Vertrauen in solche Hilfsmittel ist aber nicht immer sehr groß. Ich zeige hier, wie man mit ganz einfachen Forth-Programmstücken aus meinen bisherigen VD-Arbeiten das Problem nachvollziehbar bewältigen kann: Man lasse sich (in null-komma-nix) die den 'Magic Numbers' zugeordneten Sektoren ausgeben. Die gesuchten Anfangssektoren der Betriebssysteme (sprich, deren Bootsektoren) befinden sich darunter. Das Programm läuft - ohne irgendwelche 'Umschalter' - unter ZF genauso gut wie unter Turbo-Forth.

Schnelleinstieg

Meine Partitionstabelle ist 'zerschossen'. Ich versuche, sie zu reparieren. Alle Daten auf der Festplatte mögen noch in Ordnung sein. Viel wäre gewonnen, wenn ich an die Boot-Sektoren der diversen Betriebssysteme käme. Wo liegen sie? Eins ist sicher: Sie fangen alle mit einem Boot-Sektor (von 512 Bytes) an. Noch was ist sicher: Jeder Boot-Sektor endet (bei Offset 1fe 1ff) mit dem Byte-Paar 55 aa, den 'Magic Numbers'. Es kann ohne Weiteres auch andere Sektoren geben, die auf 55 aa enden, die aber keine Boot-Sektoren sind. Im Normalfall ist die Zahl der Kandidaten für einen Boot-Sektor aber sehr gering und auf jeden Fall überschaubar. Ich sichte meine bisherigen VD-Arbeiten zum Thema und gebe ein Forth-Programm an, mit dem ich mir hintereinanderweg die Lage der gesuchten Boot-Sektoren ausgeben lassen kann. So gerüstet, ist es (mit ein bisschen Nachschlagen in der PC-Literatur) nicht mehr allzu schwer, die verlorengegangene Partitionstabelle wiederherzustellen.

Vorgeschichte

Pünktlich zu Weihnachten (im Unglücksjahr 2013!) brach meine Anlage wieder mal zusammen. Ich hatte mir aus Versehen die Partitions-Tabelle im BIOS-MBR (ich spreche hier nicht von GPTs im UEFI) zerschossen. XP liegt bei mir auf der Festplatte neben DOS 6.2, Windows 3.11/95/98/ME und Linux 'mittendrin'. Und (viel zu) forsch war ich mit den bruchstückhaften Erinnerungsstücken über Partitionen aus meinen bisherigen VD-Arbeiten (siehe unten) ans Werk gegangen. Ergebnis: Ich hatte versäumt, ein MBR-Backup anzulegen, habe einen falschen 'Knopf' gedrückt - und schon war die Partitions-Tabelle futsch! Ich darf hier kurz berichten, wie ich mit meinen früheren Überlegungen (siehe Literaturangaben weiter unten) aus dem Sumpf von Milliarden und Abermilliarden von (möglichen, aber auch unmöglichen) Bytes mit Hilfe von Forth wieder herauskam.

Sowohl für Turbo-Forth wie auch für ZF

Die drei Worte `#line #out at` sind in ZF anders definiert als in Turbo-Forth. Das Zusammenspiel dieser Worte, in Verbindung mit einem geeigneten Einsatz des Bildschirm-Scrollings, hat mich aber in die Lage versetzt, das Programm so zu arrangieren, dass es für ZF genauso läuft wie für Turbo-Forth. Das Programm merkt unter ZF nicht, dass die genannten drei Worte hinter ihren Bezeichnungen streng genommen etwas anderes tun als in Turbo-Forth - und umgekehrt.

Es gibt sicher Leser, die in ZF und in Turbo-Forth gleichermaßen zu Hause sind. Ich überlasse es solchen Freunden der genannten beiden F83-Derivate, meinen Vorschlägen sowohl unter ZF wie auch unter Turbo-Forth nachzuspüren. Man lernt dabei eine ganze Menge über das System von Tom Zimmer einerseits und das System von Marc Petremann andererseits - zwei Systeme, die weitestgehend übereinstimmen und doch an manchen Stellen auch so ganz anders aufgezogen sein können. Die Standardisierungs-Bestrebungen in allen Ehren, aber es fällt nicht immer leicht, zwischen Forth als Sprache und Forth als Mittel der Systemgestaltung zu unterscheiden!

Partition Table Doctor

Ich hätte sagen können, das aus Russland stammende 'Partition Table Doctor 3.5' [4] schafft es. Das reicht mir als Forth-Adept aber nicht. Das sich Verlassen auf undurchsichtige Erklärungen und das durchaus nicht immer unanfechtbare Wissen anderer macht mir keinen Spaß - und kann mich auch allzu leicht in die Irre führen. Ich möchte die Dinge mir selbst (und vielleicht auch anderen) in ganz kleinen Forth-Bausteinen genauer erklären. (In den Tagen der Heimcomputer gab es die Programmchen von 1-Kilobyte Länge. Solche 'Bausteine' schweben mir hier, wie bisher in anderen VD-Arbeiten auch schon, vor.)

Der leidliche Ärger mit DOS unter XP

Ich verwende die Bezeichnung 'XP' als Kennzeichnung für das, was etwa mit XP seinen Anfang nahm und was aus dieser Richtung noch alles auf uns zukommen kann. Versuchen Sie mal, die hier zur Rede stehenden Untersuchungen von der XP-Warte aus durchzuführen! Natürlich kann man von XP aus - wenn also die Anlage von XP gebootet wurde - auf DOS zugreifen. Aber nicht auf die Festplatte direkt. Originalton XP: "Eine Anwendung hat versucht, auf die Festplatte direkt zuzugreifen. Dies wird

nicht unterstützt. Dies kann zur Folge haben, dass die Anwendung nicht korrekt funktioniert. Klicken Sie auf 'Schließen', um die Anwendung zu beenden." Eine solche Meldung steigert meine Aufmüpfigkeit ins Unermessliche! Warum soll ich denn da nicht mal schnell direkt zugreifen können? Es kommt doch ganz darauf an, was ich mit der Festplatte vorhabe! Wo ist der Schalter, der es mir erlaubt, das Verbot zu umgehen. Es hängt ja nicht jede Anlage an der Leine. Und warum soll ich nicht meine eigene Maschine nach Belieben so einsetzen können, wie ich will? Auch unter und ab XP? Ganz schlimm wird es, wenn ich mein selbstfabriziertes Programm (siehe Listing) vom XP-Arbeitsplatz aus aufrufe: Dann erhalte ich einfach schon beim ersten Sektor einen 'Lesefehler'. Ohne Vorwarnung - und ohne Angabe von Gründen. Natürlich, ich müsste mir die Mühe machen, solange herumzustudieren, bis ich einen passablen 'Abschalter' gefunden habe. Aber dazu habe ich keine Lust. Und der Stand meiner Kenntnisse reicht nicht. Soviel zum Grund, weshalb ich den Ausweg über die Boot-Diskette gewählt habe. Und der wird mir vom System nicht verboten!

Magic Numbers

(Zumindest) bei Betriebssystemen, die als abgeschlossene Partition auf FAT16/32 oder NTFS (nur immer eine einzige Partition in der Partitions-Tabelle aktiv) basieren, werden die Bootsektoren mit dem Byte-Paar 55 aa (den 'Magic Numbers') abgeschlossen. Das gilt auch für den MBR, für die EBRs und für die Bootsektoren einer erweiterten DOS-FAT16-Partition. Trotzdem: Die Zahl der zu findenden Magic Numbers ist (normalerweise - bei ansonsten nicht zerschossener Festplatte) nicht sehr hoch. Mit anderen Worten: Hat man die HD-Adressen der Sektoren, die an ihrem Ende das Magic-Number-Paar 55 aa aufweisen, gefunden, dann hat man auch einen schnellen Überblick über die gesuchten Bootsektoren (also über die vermeintlich verloren gegangenen Betriebsystem-Daten).

Ziel

Ich möchte bei einer zerschossenen Partitionstabelle die benötigten Daten soweit rekonstruieren, dass ich mein XP-System wiederfinde. Zu schaffen ist das über das Wissen darum, dass der Bootsektor der XP-Partition, wie auch der anderer Partitionen, an einer Partitionsgrenze liegen und über die Magic Bytes 55 aa an seinem Ende gekennzeichnet sein muss. Mein Wissen über diese Dinge beziehe ich u.a. aus der Wikipedia (Google sei Dank!).

Wozu?

Der Bootsektor von XP beginnt mit den Bytes EB 5B 90 4E 54 46 53 (nachprüfen oder über Google erkundigen) und endet mit den Magic Numbers (dem Bytepaar) 55 AA (ich mache im gesamten Artikel keinen Unterschied zwischen Groß- und Kleinschreibung von Zahlen). Die ersten drei Bytes (hier EB 5B 90) sind ein stereotyper Sprungbefehl (an eine bestimmte Stelle im Bootsektor, die hier nicht weiter interessiert), die nächsten vier Bytes sind die ASCII-Codes für die Buchstabenfolge NTFS. (Diese Buchstaben werden von 20 20 20 20 gefolgt. In jedem brauchbaren Bootsektor (selbst in den einzelnen logischen Laufwerken einer erweiterter DOS-Partition) findet man eine solche Buchstabenfolge von bis zu 8 Bytes.) Hat man also erst einmal die besagte Buchstabenfolge gefunden, dann kann man, wenn außer der Partitions-Tabelle nichts beschädigt wurde, ziemlich sicher sein, dass man den gesuchten Bootsektor gefunden hat. Das gilt für die Bootsektoren beliebiger Betriebsystems-Partitionen (z.B. DOS 6.2 und Windows ME). Mich interessierte hier nur Windows XP auf NTFS. Und wenn der Bootsektor eines Betriebssystems gefunden ist, dann ist ein wesentlicher Schritt für die Wiederherstellung der Partitions-Tabelle getan.

Vom Fluch der gigabyte-gewaltigen Platten

Bei 20 Gigabyte ging alles zügig. Sektor um Sektor. Bei 200 Gigabyte aber wurde meine Geduld auf eine harte Probe gestellt: Ich war zum Abbrechen gezwungen. Zum Glück wusste ich aber, wo die Partitionen so ungefähr lagen, hatte ich sie doch selbst (wenn auch 'in grauer Vorzeit') angelegt. Und außerdem kommt man mit FDISK oder Ähnlichem, ob mit DOS oder auch ohne, (auch Linux tut gute Dienste), schnell zur ungefähren Festplatten-Aufteilung. Die Meldung 'Non-DOS' oder Ähnliches spielt dabei keine Rolle: Ich brauchte nur LBAmagnum (per 2! - siehe Listing) so ungefähr vor die Stelle zu legen, wo ich das gesuchte und verloren gegangene Betriebssystem (also dessen Bootsektor) vermutete - und ab ging dann die Post: Bis zum nächsten (dem gesuchten) Bootsektor wieder 'zügig'.

Ausgangspunkt der Überlegungen

Was heißt 'zerschossene Partitions-Tabelle'? Ich darf 'Tabula rasa' machen und für die Zwecke des vorliegenden Artikels (vorsichtshalber) von einer Partitionstabelle (nicht vom ganzen MBR - nur von den 4 Zeilen zu je 16 Bytes der Partitionstabelle) ausgehen, bei der alle Bytes auf 0 gesetzt sind (Bytes 1be bis 1fd = 0). Was sollte also (in diesem Bericht) auch nach dem Zerschossensein noch bekannt sein? Man weiß, dass die Partitionstabelle im Sektor 0 (LBA) der Bootplatte gelegen haben muss - und man weiß, dass der MBR (der Sektor 0) als letzten Markierungspunkt die Bytes 1fe = 55 und 1ff = aa tragen muss. Das lässt sich also schon einmal (zur Rekonstruktion der Partitionstabelle) eintragen. Außerdem soll die Rede nur von einer 'zerschossenen Partitionstabelle' sein - nicht von einem Zerschossensein der (gesamten) Festplatte.

Was kann als noch gesichert gelten?

Bootsektoren, der MBR, die EBRs sind normalerweise mit dem Sektor-Offset (1fe)=55, (1ff)=aa abgeschlossen. Daran möge sich nichts geändert haben! Zur Aufgabe habe ich mir nur gestellt, solche Stellen (auf der Bootplatte), an denen das Magic-Number-Bytepaar 55 aa auftritt,

der Reihe nach aufzusuchen und für die weitere Analyse vorzubereiten. (Der 'Partition Table Doctor 3.5' [4] kann das prima. Aber warum soll ich mich auf ein von irgendwem stammendes Programm verlassen, wenn ich diese (kleine) Frage schon mit wenigen Forth-Mitteln für mich in durchsichtiger Weise selbst erledigen kann? So sei es!

Was bleibt als mein ureigenstes Ziel?

Ich weiß, dass ich den verlorenen gegangenen Zugriff auf die XP-Partition wiederbekomme, wenn ich der Partitions-Tabelle folgende Daten (als Offset im LBA-Sektor 0 der Bootplatte) wiederanbieten kann:

00 = 80 (als Kennzeichen der - als einzige anzunehmenden - Bootpartition) 02 = 0 (als Tatsache, dass da (bei mir) kein MBR 'dazwischenliegt'

Und dann bleibt da noch die Aufgabe, aus den bis dato schon wiederbereitgestellten Daten den 'Rest' der XP-Zeile in der Partitionstabelle wiederzugewinnen. Der 'Partition Table Doctor' 3.5 [4] kann das prima. Ich will versuchen zu zeigen, wie ich es mit meinen in früheren VD-Artikeln als Bausteine bereitgestellten Forth-Mitteln ebenfalls schnell und nachvollziehbar schaffe.

Aber gemach!

Ich habe kürzlich die Bemerkungen unseres FG-Freundes Erich Wälde (als Redakteur) im VD-Heft 1/2014 (ein Lob seinem ständigen Einsatz für die FG!), gelesen, nämlich dass es nicht darum gehe, ein fertiges Produkt zum code-getreuen Abkupfern als VD-Artikel anzubieten, sondern darum, Anregungen für die geneigten Leser und Innen zu liefern. Also liefere ich hier mal schnell einen 'Baustein', einen (vergleiche Titel) solchen zum Auffinden der Magic Numbers - für den Fall, dass man voller Verzweiflung kurz davor stehen sollte, alles aufzugeben, und an die berühmtberüchtigte 'Neuinstallation' denkt. Den Baustein liefere ich natürlich in Forth!

Nicht alles auf einmal

Ich muss bei meinem Vorhaben alle Sektoren nacheinander auf 55 aa an ihren Enden untersuchen. Es wäre verhältnismäßig leicht (Aufgabe für die Leser!) auch die Untersuchung auf XP-oder-nicht-XP mit einzubeziehen. Ich darf mich aber damit begnügen, den XP-Ball an die Leser weiterzugeben, und konzentriere mich hier nur auf das (interaktiv aufgezogene) Auffinden der 'Magic Numbers'.

Voraussetzungen

Ich gehe von einem DOS-6.2-System auf einer 3,5-Zoll-Diskette aus, das ich auf einem USB-Diskettenlaufwerk laufen lasse. (Damit ist also auch ein Betrieb auf einem diskettenlosen PC (z.B. Laptop) möglich.) Die Diskette soll so wenig DOS-Teile enthalten, dass darauf Turbo-Forth und die neu eingeführten Programmteile meines diesmaligen Artikels (sie sind alle im Listing enthalten) laufen können. Es ist insbesondere nicht nötig, dass das Forth-auf-DOS-System auf der Festplatte läuft. Es soll aber eine PC-kompatible Festplatte (mit BIOS) enthalten sein, auf die das Disketten-DOS anstandslos zugreifen kann (LBA-Zugriff über den erweiterten Interrupt 13h). Man stelle sich eine Platte mit mehreren Windows-Systemen - aber gegebenenfalls auch ohne DOS - vor.

Ich habe das Ganze neben DOS-6.2 auch unter FreeDOS von Diskette getestet. FreeDOS kann man sich aus dem Internet über ein DiskImage in Verbindung mit Rawwrite besorgen. Auf FreeDOS lasse ich ebenfalls Turbo-Forth in 16-Bit-Ausführung laufen. (Wegen Turbo-Forth konsultiere man den amerikanischen FTP-Server taygeta.com oder dessen Mirror in Bremen. Vielleicht findet man auch den Zugang zum Turbo-Forth-Autor Marc Petremann in Frankreich.)

Es ist klar, dass das gesamte Listing des vorliegenden Artikels bei Weitem weniger Platz benötigt, als auf eine 3,5-Zoll-Diskette passt. Man kann also auch noch weitere Hilfsmittel (z.B. PC-TOOLS und einen Schnell-Editor) mit auf die Diskette laden. Gebootet wird (für die Reparatur-Arbeiten mit dem System aus meinem Artikel) über die Diskette.

Hat man die Festplatte wieder auf Vordermann gebracht, dann stellt man das Booten natürlich wieder auf das gewünschte Betriebssystem - eben auf XP - um.

Das Arbeiten mit FreeDOS hat den Vorteil, dass man darüber auch den Zugriff auf USB-Sticks hat. Von Diskette! Und wenn man (wie ich) dennoch auf der Festplatte eine DOS-Partition eingerichtet hat, dann kann man die hier angedeuteten Arbeiten (siehe Listing) auch über DOS 6.2 (oder andere DOS-Versionen) ausführen - soweit man den Motto-Hairu-Treiber von Panasonic (siehe Google) einsetzt und die maximale Dateigröße von 2 GB bei FAT16-Partitionen beachtet. (Immerhin sind 2 GB ein klein wenig mehr als 1,44 Megabyte ;-)

Endgültiges Ziel

Ich möchte die dem XP-System zugeordnete Zeile der Partitions-Tabelle soweit wieder herstellen, dass ich danach durch Booten der Festplatte (der Festplatte, nicht des Disketten-Laufwerks) XP aufrufen kann.

Nahziel

ist dabei das Auffinden der dem Bootsektor von XP zugeordneten Magic Numbers.

DOS unter XP?

Aus reiner Neugierde wollte ich vom schließlich wiederhergestellten XP-System booten und Turbo-Forth mit dem hier wiedergegebenen Zusatz-System, das auf der Diskette liegt, per include aufrufen. Forth funktionierte dabei voll (ich kann beispielsweise : xxx dup ; ausführen). Aber sobald ich mit irgendeinem Baustein aus dem Turbo-Forth-System an die Festplatte heranwill (z.B. über getmbr [ret] aus [2]), versagt der Vorgang (siehe oben). Nun ja, also bleibe ich dabei, meine Vorhaben

über eine Bootdiskette in Bewegung zu setzen! Nebenbei gesagt, gelingt es mir auch nicht, aus XP heraus das ansi.com aus [1] zur Mitarbeit zu bewegen.

Andererseits rührt das DOS-Programm - ob von Diskette oder von einer FAT-Partition - das XP-System nicht an. Es ignoriert es einfach. Aber der (sektorweise) Zugriff auch auf die Teile der Festplatte, die das XP-System enthalten, ist natürlich ohne Weiteres möglich. Das ist der Grund, weshalb ich (für die Absichten des vorliegenden Artikels) auf den Ausweg über die Boot-Diskette verfallen bin.

Auch anderswo Behinderung durch das Zugriffsverbot

Ich habe zu verschiedenen Zeiten immer wieder mal zu CWDSKEDT [5] gegriffen, um Dateien zu reparieren - und mich immer wieder über den als Fortschritt verkauften Rückschritt geärgert. Ruft man in einem XP-gebooteten System CWDSKEDT.EXE auf, dann erhält man die lakonische Auskunft: “Im Moment wird Microsoft Windows NT ausgeführt. Zugriff auf Festplatten ist nicht möglich.”

Bedienungsanleitung

Der vorliegende Artikel greift sich mit den “Magic Numbers” nur einen eng begrenzten Bereich der PC-Fehler-Analyse heraus. Dennoch enthält er bereits ein ganzes Paket von Dingen, die man sich im Einzelnen kaum merken kann oder will. Am schnellsten verschafft man sich einen Überblick, wenn man magnum? aufruft und dann in der erscheinenden Auswahlzeile ’Raus’, also [Esc], drückt.

Hat man sich dem Ende der Festplatte genügend genähert (bei großen Festplatten unter Mitwirkung von Überlegungen, die ich hier nicht weiter verfolgen möchte) und erreicht man schließlich das Ende, dann erscheint die Meldung ’Sektor-Lesefehler’.

Nach getaner Arbeit möchte man sicher gern den Master Boot Record aufrufen, um dort die gefundenen Veränderungen einzutragen. Das erledigt sich über getmbr [ret]. Das Abändern einzelner Bytes gehört zum Standard-Repertoire eines jeden Forth-Systems und braucht hier nicht erklärt zu werden. Das Abändern selbst vollzieht sich im RAM-Puffer ab Adresse sectbuf.

Bei den im vorliegenden Artikel aufgelisteten Forth-Worten habe ich vorsichtshalber darauf geachtet, nur Lesebefehle aufzulisten, keine Schreibbefehle.

Literatur

1. Mefford, Michael, J.: PC Magazine (1988). `http://www.pcmag.com/article2/0,2817,5343,00.asp`
2. Behringer, Fred: Teil 1 meiner Artikelserie in der Vierten Dimension (3/2008) `http://forth-ev.de/filemgmt/viewcat.php?cid=44`
3. Behringer, Fred: Teil 3 meiner Artikelserie in der Vierten Dimension (1/2009). `http://forth-ev.de/filemgmt/viewcat.php?cid=52`
4. Partition Table Doctor 3.5. der PTDD Group. Live-CD aus der Begleit-DVD von c’t 24/2008. `http://www.ptdd.com.`
5. Walter, Christoph: CWDSKEDT.EXE 2.22 (46 KB). Diskeditor für DOS. Freeware.

Listings

Magic Numbers

```
1
2   \ Listing
3   \ -------
4
5   \ ******************************************
6   \ *    Name: MAGICNUM.FTH                  *
7   \ * Aufgabe: "Magic Numbers" bei zerschosse- *
8   \ *          ner  Partitions-Tabelle  suchen *
9   \ *          und Bootsektor (von XP) auf der *
10  \ *          Boot-Festplatte wiederauffinden *
11  \ * Sprache: Turbo-Forth oder ZF -  (16-Bit) *
12  \ *   Autor: Fred Behringer                  *
13  \ *          Forth-Gesellschaft              *
14  \ *          01. 06. 2014                    *
15  \ ******************************************
16
17  \ Das Programm laeuft auch unter ZF von Tom Zimmer bestens.
18
19  \ Die Forth-Worte 1+! und (cursor) gibt es in ZF nicht. Ich habe sie durch
20  \ solche Worte zusammengestueckelt, die man in beiden Systemen findet.
21
22  \ Arbeitet man mit einer Anlage, die kein ANSI.SYS in der CONFIG.SYS zur
23  \ Verfuegung hat, dann tut es auch das ANSI.COM von Michael J. Mefford [1].
```

```
24
25  \ Das Programm moege den (willkuerlichen) Namen magicnum.txt tragen. Dann
26  \ Turbo-Forth: forth include magicnum.txt
27  \          ZF:    zf fload   magicnum.txt
28
29
30  hex             \ Alle Zahleneingaben sind hexadezimal zu verstehen!
31
32
33  \ Die folgenden Worte stammen aus frueheren VD-Arbeiten von mir und werden
34  \ hier ohne Kommentare und mit nur wenigen Formatierungen wiedergegeben.
35
36  210 allot here here 0f and - 200 - constant sectbuf
37
38  \ sectbuf = Anfangsadresse des Sektorpuffers
39
40  \ Sektor lesen: cx = Spur/Sektor-Kombination
41
42  \ cx = Bits F-0 = FEDCBA98 76543210
43  \    =               ch       cl
44
45  \ Hierbei ist:
46  \ Spur = 76FEDCBA98  und  Sektor = 543210
47  \        76(cl) &ch               von cl
48
49  code (getsect) ( seite spur/sektor -- )    \ Bessere Aufbereitung in [2]
50       ds push es pop 80 # dl mov ( 80 = erste Festplatte )
51       cx pop ax pop al dh mov sectbuf # bx mov
52       201 # ax mov 13 int next end-code     \ 201 # = Lesen
53
54  \ Die Partitionstabelle beginnt bei Adr-Offset 1be der Boot-Platte. Der MBR,
55  \ aber auch die EBRs und die Bootsektoren enden mit den Magic Numbers 55 aa.
56  \ 10 Bit Spur und 6 Bit Sektor --> 16 Bit Spur/Sektor. So steht es im
57  \ Master-Boot-Record und so wird es in Int 13h, 2/3 in cx verlangt.
58
59  code sp,sc>spsc ( sp sc -- spsc )          \ Kommentare in [2]
60       ax pop 3f # ax and bx pop 6 # cl mov bh cl shl
61       bh al or bl ah mov ax push next end-code
62
63  \ Umkehrung von sp,sc>spsc.
64  code spsc>sp,sc ( spsc -- sp sc )          \ Kommentare in [2]
65       ax pop ax bx mov 6 # cl mov bl cl shr bl dh mov
66       ah dl mov dx push 3f # ax and ax push next end-code
67
68  \ MBR der ersten Platte lesen und nach sectbuf speichern.
69  : getmbr ( -- ) 0 1 (getsect) ;
70
71  : showsectbuf    ( -- ) sectbuf 200 dump ; \ Puffer sectbuf anzeigen
72  : showsectbuf100 ( -- ) sectbuf 100 dump ; \ Nur 100 Bytes anzeigen
73
74  \ ========================================================================
75
76  \ Was im vorliegenden Artikel eigentlich interessiert, ist die Entwicklung
77  \ eines Forth-Wortes, mit dessen Hilfe man sich Sektoren auch von groesseren
78  \ Festplatten ansehen kann, Sektoren also, bei denen die CHS-Adressierung
79  \ versagt und bei denen man auf LBA-Adressierung zurueckgreifen muss. Alles,
80  \ was diesem Ziel dient, kann im dritten Teil meiner Serie nachgelesen werden
81  \ [3].
82
83  \ Bei der LBA-Schreibweise der Sektoren geht es mir (in Bezug auf die Magic
84  \ Numbers) nur um das Auffinden. Das Schreiben kommt im vorliegenden Artikel
85  \ nicht vor. Im Uebrigen verweise ich auf [3] und gebe die benoetigten Worte
86  \ aus [3] ohne Kommentare und ohne spezielle Formatierungen wieder.
87
88  \ Zunaechst benoetige ich den Puffer dap aus dem dritten Teil meiner Serie
89  \ [3] zur Parameter-Uebergabe. Ich wiederhole die dortigen Angaben:
90
91  28 allot here here 0f and - 18 - constant dap    \ Anfang des dap-Puffers
92
93  \ Ich brauche eine schnelle Moeglichkeit, an den Wert cs heranzukommen. Das
94  \ wollte ich in Turbo-Forth ueber die dortige Konstante dsegment machen. In
95  \ ZF gibt es kein dsegment, aber auch keinen direkten Zugang zu cs. Ich
96  \ fuehre fuer beide Systeme die Konstante csegment ein. In Turbo-Forth stoert
97  \ das nicht.
98
99  code csegment ( -- cseg ) cs push next end-code
```

```
100
101  \ LBA-Sektor sl (32 Bit!) von der Boot-HD (hier immer 80) nach Puffer sectbuf
102  \ holen. fl <> 0 bedeutet Fehler. (Auf Fehler gehe ich hier aber nicht ein.)
103
104  code getLBAsect ( sl -- fl )         \ sl doppeltgenau; Fehler, wenn fl <> 0
105       18 # dap #) mov 1 # dap 02 + #) mov sectbuf # dap 04 + #) mov
106       csegment # dap 06 + #) mov 80 # dl mov ax pop ax dap 0a + #) mov ax pop
107       ax dap 08 + #) mov 0 # dap 0c + #) mov 0 # dap 0e + #) mov si push
108       dap # si mov 4200 # ax mov 13 int si pop ah al mov ax push next end-code
109
110  \ Achtung: Wenn man mit getLBAsect nicht gleich beim ersten Versuch zum Ziel
111  \ kommt, dann ist es so gut wie sicher, dass man den Wert sl nicht als
112  \ "Punktzahl", sondern nur einfachgenau angesetzt hat!
113
114  \ ============================================================================
115
116  \ Soweit die hier verwendeten Worte aus frueheren Artikeln.
117  \ Die folgenden Worte sind neu im vorliegenden Artikel.
118
119  \ CHS -- LBA: Das im vorliegenden Artikel eingesetzte Turbo-Forth ist ein
120  \ 16-Bit-System. Zum Ausprobieren wurde eine Festplatte von 200 GB verwendet.
121  \ Zur Ausschoepfung aller erreichbaren Adressen kommt man dabei mit 16 Bit
122  \ (CHS) nicht durch. Zum Einsatz der LBA-Adressierung benoetigt man 32-Bit.
123  \ Mit anderen Worten: Auf das 16-Bit-Turbo-Forth muss die 32-Bit-Arithmetik
124  \ (also doppelt-genaue Zahlen, sprich "Punktzahlen", und doppelt-genaue
125  \ Operationen) angewandt werden. Entsprechendes gilt fuer ZF.
126
127  \ Zur Erinnerung an den Punkt in den Punktzahlen darf ich das an sich nicht
128  \ besonders wichtige (und ueberaus unuebliche) folgende Wort einfuehren:
129
130  : d.. ( d -- ) (d.) type ." " ;       \ Punktzahl-Ausgabe mit Punkt
131
132  2variable LBAmagnum                   \ Aktuelle lineare Sektor-Adresse (LBA)
133
134  : weiter? ( -- )                      \ Fortfahren oder Abbrechen?
135       0 #line @ at 4a spaces           \ Cursor auf Spalte 0, 4a Bytes Platz,
136       0 #line @ at                     \ Cursor wieder auf Spalte 0.
137       ." Weiter (ab " LBAmagnum 2@ 1.  \ Dann Ausgabewert nach "ab wann"
138          d+ d.. ." ): "                \ mit Punkt versehen
139       ." [Ret] ------ Raus "           \ und Meldung "Weiter/Raus"
140       ." (dann Neueingabe?): "         \ ausgeben.
141       ." [Esc]" 5 spaces ;
142
143  variable zaehler1                     \ Hochzaehlen und dann Punkt ausgeben
144  variable zaehler2                     \ Wenn zaehler2 voll, dann von Neuem
145
146  : verlauf                             \ Wartepunkte ausgeben.
147       zaehler1 dup @ 1+ swap !         \ Zaehler1 um 1 weiterschalten und
148       zaehler1 @ 80 =                  \ Punkt ausgeben. Bei Erreichen von 80:
149       if 0 zaehler1 ! ." ."            \ Zaehler1 auf 0 setzen. Punkt ausgeben.
150         zaehler2 dup @ 1+ swap !       \ Zaehler2 um 1 weiterschalten.
151       then
152       zaehler2 @ 20 =                  \ Bei Erreichen von 20 den zaehler2
153       if 0 zaehler2 !                  \ auf 0 setzen.
154         20 #line @ at 2a spaces        \ Cursor auf Spalte 20, 2a Bytes Platz,
155         20 #line @ at                  \ Cursor wieder auf Spalte 20.
156       then ;
157
158  \ Achtung: Man beachte die Parameter 80, 20, 20, 2a, 20.
159  \          Ueber diese lassen sich die Wartepunkte beliebig anders verteilen.
160
161  variable cfa-1-weiter                 \ Fuer tick(cfa) (defer ueber Variable)
162
163  : weiter ( -- )
164       0 #line @ at 4a spaces           \ Cursor auf Spalte 0, 4a Bytes Platz
165       0 #line @ at                     \ schaffen, Cursor wieder auf Spalte 0.
166       LBAmagnum 2@ 1. d+ LBAmagnum 2!  \ LBAmagnum um 1. weiterruecken und ab
167       LBAmagnum 2@                     \ da naechsten Sektor mit Magic Number
168       #line @ 16 > if dark then        \ suchen. (Vorher pruefen, ob
169       cfa-1-weiter perform ;           \ kuenstliches Scrollen noetig.)
170
171  : forth-prompt ( -- )
172        0 #line @ at                    \ Cursor auf Spalte 0 setzen
173       ." Das System befindet sich "    \ und Erklaerungen auf dem
174       ." jetzt im Forth-Prompt.  "     \ Bildschirm ausgeben.
175       ." Moegliche Neueingaben:"  cr
```

```
176       ." xxxxxxxx. getnextmagnum  "
177                           29 spaces
178       ." [ret]"                   cr
179       ." magnum? "          3a spaces
180       ." [ret]"                   cr
181       ." weiter "           3b spaces
182       ." [ret]"                   cr
183       ." LBAmagnum 2@ d.." 32 spaces
184       ." [ret]"                   cr
185       ." showsectbuf "      36 spaces
186       ." [ret]"                   cr
187       ." showsectbuf100 "   33 spaces
188       ." [ret]"  ( 2a spaces )    cr
189       ." Einen beliebigen Sektor "
190       ." (auch solche ohne 55 aa) "
191       ." bekommt man wie folgt:"  cr
192       ." xxxxxxxx. getLBAsect "
193       ." showsectbuf (oder "
194       ." showsectbuf100)"  0c spaces
195       ." [ret]    " ;
196
197   : getnextmagnum ( d1 -- )              \ d1 = doppeltgenau (Zahl mit Punkt)
198       0 zaehler1 !  0 zaehler2 !         \ Beide Verlaufszaehler initiieren.
199       2dup 0. d<
200       if     cr cr ." Negative Sektor-Adressen unzulaessig. "
201                ." Eingabe wiederholen!" cr drop drop exit
202       then
203       20 #line @ at                      \ Cursor auf Spalte 20 setzen.
204       begin
205         verlauf                          \ Fortschrittspunkte ausgeben.
206         2dup LBAmagnum 2! 2dup           \ (LBAsect) nach LBAmagnum speichern.
207         getLBAsect                       \ (LBA-Sektor) nach sectbuf speichern.
208         0 <>                             \ Fehler?
209         if                               \ Ja,   dann Sektor-Adresse d. (soweit
210           cr ." Sektor-Lesefehler! "     \       gueltig) nach LBAmagnum
211           ." Neue Eingabe machen!" cr    \       speichern und Bereitschaft fuer
212           LBAmagnum 2!  exit             \       neuen Anfang anzeigen.
213         else                             \ Nein, wenn kein Lesefehler, pruefen,
214           #line @ 16 > if dark then      \       ob kuenstliches Scrollen
215           sectbuf 1fe + @ aa55 =         \       noetig. Ansonsten Weitersuche.
216           if 2dup                        \   Ja, dann nach LBAmagnum speichern,
217              LBAmagnum 2! 3e #line @ at  \       Sektor-LBA-Adresse
218              8 d.r ." ."                 \       rechtsbuendig ausgeben. Sodann
219              cr weiter?                  \       Abfrage: Weiter oder Raus?
220           \ ------------------------------------------------------------------
221           begin key dup
222             1b = if forth-prompt         \     Wenn Escape gedrueckt,
223                     drop exit            \          dann  Raus-Meldung ausgeben
224                  then                    \          und zum  Forth-Prompt gehen.
225             0d = if weiter               \     Wenn Return gedrueckt,
226                     true                 \          dann die Magic-Number-Suche
227                  then                    \          ueber until=true fortsetzen.
228           until                          \     Sonst zurueck zu begin.
229           exit                           \     true --> exit zur Weitersuche.
230           \ ------------------------------------------------------------------
231           else 1. d+                     \ Nein, dann LBA-Zaehler um 1 (32 Bit)
232              0                           \       weitersetzen.  Naechste until-
233           then                           \       Ebene ueber die 0 einleiten.
234         then                             \ Sonst raus: Magic-Sektor jetzt ueber
235       until ;                            \ showsectbuf oder ...buf100 aufrufbar.
236
237   ' getnextmagnum cfa-1-weiter !         \ "Tick" zur Belegung der Variablen cfa-1-weiter.
238
239   : magnum? ( -- )
240       0. getnextmagnum ;
241
242   \ Ende des eigentlichen Programms
243
244   \ Und jetzt noch schnell eine Beispiels-Ausgabe.
245   \ Begonnen wird dabei mit der durch [ret] abgeschlossenen Eingabe magnum?
246   \ Die Punkte versichern dem Benutzer, dass noch alles laeuft.
247   \ Nach jeder Zeile wird (hier im Beispiel) [weiter ret] eingegeben.
248   \ Nach 956E. (willkuerlich) wird in diesem Beispiel [raus ret] eingegeben.
249   \ Daraufhin erfolgt die Restausgabe.
250   \ (Zwischen Gross- oder Kleinbuchstaben wird nicht unterschieden.)
251
```

```
252  \ Und hier das Beispiel:
253  \ ------------------------------------------------------------------------
254  \ magnum?                                                              0.
255  \                                                                      F.
256  \                                                                     3F.
257  \                                  ............                      656.
258  \                                  .....                            78FD.
259  \                                  ........................         956E.
260  \ Das System befindet sich jetzt im Forth-Prompt.  Moegliche Neueingaben:
261  \ xxxxxxxx. getnextmagnum                                            [ret]
262  \ magnum?                                                            [ret]
263  \ weiter                                                             [ret]
264  \ LBAmagnum 2@ d..                                                   [ret]
265  \ showsectbuf                                                        [ret]
266  \ showsectbuf100                                                     [ret]
267  \ Einen beliebigen Sektor (auch solche ohne 55 aa) bekommt man wie folgt:
268  \ xxxxxxxx. getLBAsect showsectbuf (oder showsectbuf100)             [ret]
269  \ ------------------------------------------------------------------------
270
271  \ Und hier das zusammengefasste Glossar (fruehere und neue Forth-Worte):
272
273  \ sectbuf                           \ Anfangs-Adresse des Sektorpuffers
274  \ (getsect) ( seite spur/sektor -- ) \ Sektor in den Sektorpuffer holen
275  \ getmbr                            \ MBR in den Sektorpuffer holen
276  \ showsectbuf                       \ Puffer sectbuf anzeigen
277  \ showsectbuf100                    \ Nur die ersten 100 Bytes anzeigen
278  \ dap                               \ Anfangs-Adr des dap-Puffers (32 Bit)
279                                      \ Ab hier neue Forth-Worte:
280  \ csegment ( -- cseg )              \ Hommage an ZF
281  \ getLBAsect                        \ LBA-Sektor (LBA = 32 Bit) nach sectbuf
282  \ LBAmagnum                         \ Lineare (32 Bit) Sektor-Adresse (LBA)
283  \ d..                               \ Kosmetik: Punktzahl-Ausgabe mit Punkt
284  \ weiter?                           \ Fortfahren oder Abbrechen?
285  \ zaehler1                          \ 16 Bit. Hochzaehlen und Punkt ausgeben.
286  \ zaehler2                          \ 16 Bit. zaehler2 voll, dann von Neuem
287  \ verlauf                           \ Wartepunkte fuer Ungeduldige
288  \ cfa-1-weiter                      \ Zur Aufnahme der cfa von getnextmagnum
289  \ weiter                            \ Naechstes Magic-Number-Paar aufsuchen
290  \ forth-prompt                      \ Zum Forth-Prompt mit Erklaerungen
291  \ getnextmagnum                     \ Naechsten Sektor mit 55 aa aufsuchen
```

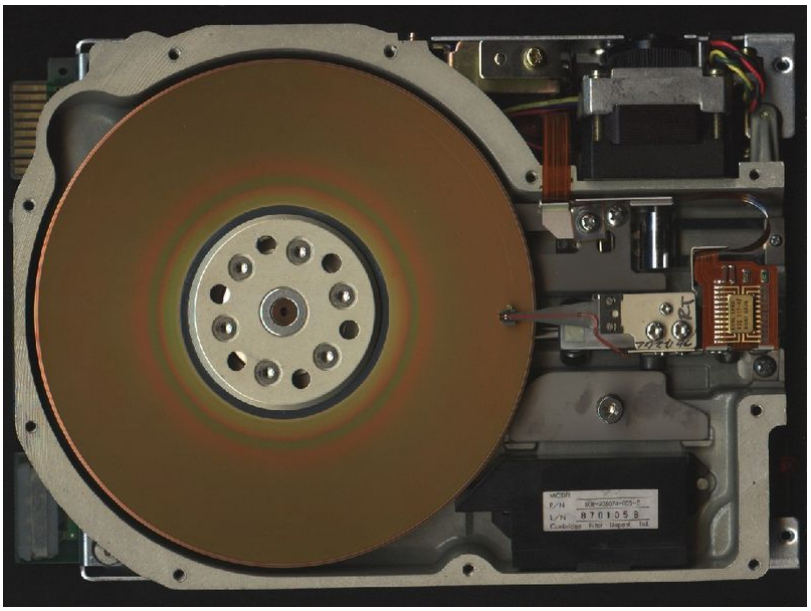


Abbildung 1: Seagate ST 506 HDD (Quelle: `https://en.wikipedia.org/wiki/ST-506`)

Grafik für Gforth Teil 2

Hannes Teich

Im Heft 1/2014 ging es um das Erstellen einer Datei zum Füttern eines Android–Tablets, so dass dieses als "Plotter" dienen kann. Die solches bewerkstelligende "App" war in Mobile Basic geschrieben und im Heft abgedruckt. Die verwendeten Kommandos sollen hier durch weitere nützliche ergänzt werden.

Motivation

Relativ harmlos ist *RNDRECT*, ein Rechteck mit gerundeten Ecken. Zusätzlich zu den Parametern für `RECT` wird der Radius der Ecken angegeben.

Zuweilen wünscht man sich statt eines ganzen Kreises ein Kreissegment. Das lässt sich mit dem Kommando *ARC* herstellen. `ARC` bezieht sich zwar nicht auf `CIRCLE`, sondern auf `OVAL`, aber der Kreis ist bekanntlich ein Sonderfall der Ellipse. Man gibt Position und Größe des unsichtbaren Rechtecks an, in das die Ellipse eingebettet ist. Bei `ARC` wird zusätzlich die Startposition des Ellipsen-Segments und die Bogenlänge angegeben. Bei `OVAL` gibt es einen Parameter *f*, der über "hohl" oder "ausgefüllt" entscheidet. Auch `ARC` hat diesen Parameter; es werden die Enden des Segments durch eine Gerade verbunden und die so entstehende Figur gefüllt.

Sehr ähnlich verhält sich das Kommando `PIE`, das die Enden des Segments – auch bei der hohlen Figur – mit dem Ellipsen–Mittelpunkt verbindet. Das entstehende "Tortenstück" lässt sich durch Parameter *f* (mit vorgewählter Farbe) füllen.

Der Parameter *f*, obgleich zum Füllen der Rechtecke, Kreise, Ellipsen und Segmente recht nützlich, kann nicht darüber hinweg trösten, dass in *Mobile Basic* weder eine allgemeine Funktion zum Füllen geschlossener Figuren vorhanden ist, noch eine Möglichkeit, gesetzte Pixel zu lesen. Schmerzlich vermisst habe ich das füllbare Dreieck, darum wird es hier mit Kommando *TRIANG* beigesteuert. Was es tut, ist leicht zu erraten; auf den Aufwand hinter den Kulissen komme ich gleich zurück.

Weiter gibt es das Kommando *ROTATE*, das alles Folgende vom *ORIGIN* aus um einen wählbaren Winkel dreht. Die vorherige Ausrichtung lässt sich mit *SAVE* speichern und mit *RESTORE* wieder herstellen.

Mit *FONT* kann einer von drei Zeichensätzen (*Monospace*, *Sans-Serif*, *Serif*) sowie seine Höhe gewählt und mit *TEXT* angewendet werden. `TEXT` erwartet die Ortsangabe des Textstarts, die Zeichenkette sowie eine Endekennung. Ob die Schrift auf allen Android-Geräten so "hohl" aussieht wie auf meinem Tablet Nexus–7 von ASUS, weiß ich nicht; auf dem Smartphone Nexus–5 von LG sieht sie jedenfalls ebenso aus.

Das Triangel–Problem

Beim Grübeln über die Möglichkeit, ein gefülltes Dreieck zu realisieren, kam ich zunächst auf die Idee, mit `PIE` ein weißes "Tortenstück" zu zeichnen und ihm dann die Rundung mit einem schwarzen `ARC` zu rauben.

Diese Lösung hat aber Nachteile: Zwei Eckpunkte des Dreiecks können angegeben werden, der dritte Punkt ist mit vertretbarem Aufwand nicht frei wählbar, sondern muss empirisch ermittelt werden. Immerhin muss das die Ellipse umschließende Rechteck geformt und passend gedreht werden, um den gewünschten Punkt auf der Ellipse zu positionieren. Abb. 2 illustriert den Vorgang, den ich zwar zum Laufen brachte, aber wieder verwarf. Ein weiterer Schönheitsfehler liegt darin, dass das schwarze `ARC` löscht, was möglicherweise vorher als Hintergrund vorhanden war.

Die Triangel–Lösung

Als besser erwies es sich, das Dreieck linienweise auszumalen. Gesetzt der Fall, Eckpunkt A sei der höchste und Eckpunkt C der tiefste: Dann wird Zeile für Zeile auf der Geraden A–C begonnen und jeweils auf A–B bzw. B–C geendet. Die Höhe des Knickpunkts B ist bekannt; sobald sie erreicht ist, wird von A–B nach B–C gewechselt. Die Reihenfolge der drei Eckpunkte ist bei der Eingabe beliebig; sie werden so oft vertauscht, bis A oben und C unten ist.

Diese 20 Kommandos stehen nun zur Verfügung

COLOR	$AA01	<red>	<gre>	<blu>	<alp>			
CLEAR	$AA02							
LINE	$AA03	<x1>	<y1>	<x2>	<y2>			
RECT	$AA04	<x>	<y>	<wid>	<hig>	<f>		
OVAL	$AA05	<x>	<y>	<wid>	<hig>	<f>		
CIRCLE	$AA06	<x>	<y>	<rad>	<fil>			
PLOT	$AA07	<x1>	<y1>	<x2>	<y2>	...	...	$AA00
PAUSE	$AA08	<sec>						
SHOW	$AA09							
END	$AA0A							
ARC	$AA0B	<x>	<y>	<wid>	<hig>	<beg>	<arc>	<f>
PIE	$AA0C	<x>	<y>	<wid>	<hig>	<beg>	<arc>	<f>
FONT	$AA0D	<typ>	<hig>					
TEXT	$AA0E	<x>	<y>	a b	c d	...	$AA00	
ORIGIN	$AA0F	<x>	<y>					
ROTATE	$AA10	<arc>						
RNDRECT	$AA11	<x>	<y>	<wid>	<hig>	<rad>	<f>	
TRIANG	$AA12	<x1>	<y1>	<x2>	<y2>	<x3>	<y3>	<f>
SAVE	$AA13							
RESTORE	$AA14							

Ein Plot wird formuliert und ausgeführt

Unten folgt das Gforth–Listing (nur was neu ist!) zum Erzeugen der Steuerdatei für das, was in Abb. 1 zu sehen ist. Auf ein Abdrucken des eigentlich interessanteren Basic-Programms wird hier im Forth-Heft verzichtet, da wurde im Heft 1/2014 schon ausreichend gefrevelt. Beide Listings (*Basic* und *Gforth*) lassen sich an bekanntem

[1] `http://forth-ev.de/filemgmt/visit.php?lid=546`

Orte zapfen[1]. Wie erwähnt verwende ich als Plotter das Tablet Nexus-7 mit Full–HD–Auflösung (1820 x 1080 Pixel) und als Werkzeug, um die Plot–Datei vom PC in den Android–Ordner `/storage/emulated/0/gforth/home/` zu übertragen, die famose App *wifi file transfer pro* von *smarterDroit*. Der Ordner wird automatisch eingerichtet, wenn man *Gforth for Android* lädt. Zum Kopieren und Verschieben von Dateien unter Android finde ich den *Cobi File Manager* praktisch, denn er zeigt zwei Fenster (für Quelle und Ziel) und dokumentiert deutlich, wo im Baum der Verzeichnisse man sich gerade befindet.

Die Basic-Dateien sind im Ordner `/storage/emulated/0-/Android/data/com.mobilebasic.FullVersion/files` zu finden, wobei darauf zu achten ist, `Android` nicht mit `android` zu verwechseln. Ich verwende nicht die "Lite"–, sondern die "Full"–Version des Mobile Basic, die gegen ein geringes Entgelt zu haben ist.

Die Bilder 1 und 2 sind etwas nachbearbeitet worden, da die Linien in Full–HD–Auflösung sehr zart ausfallen und möglicherweise beim Druck untergehen.

Zur Abarbeitung der Plot–Datei spielt es natürlich keine Rolle, woher sie kommt. Das Basic–Programm ist selbst imstande, sie zu generieren; eine mit Gforth erzeugte Datei kann vom Desktop oder vom Android–Gerät selbst kommen. (Man suche dazu den Schalter "intern" im Basic–Listing.) Mit etwas Übung ist der Weg vom Editieren zum Plotten recht flott zu nehmen. Am nützlichsten dürfte es sein, wenn ein einmal eingerichtetes Layout mit unterschiedlichen Parametern gefüttert wird.

Hier also die neuen Kommandos und der Aufruf zur Bildung von Abb.1:

```
1
2   \ Diese 10 Kommandos sind schon beschrieben worden:
3   : _begin  ( -- )  ... ;
4   : _color  { r g b a -- } ... ;
5   : _clear  ( -- ) ... ;
6   : _line  { x1 y1 x2 y2 -- }  ... ;
7   : _rect  { x y w h f -- }  ... ;
8   : _oval  { x y w h f -- }  ... ;
9   : _circle  { x y r f -- )  ... ;
10  : _plot-on  ( -- )  ... ;
11  : _plot  { x y }  ... ;
12  : _plot-off  ( -- )  ... ;
13  : _pause  { s -- }  ... ;
14  : _show  ( -- )  ... ;
15  : _end  ( -- )  ... ;
16
17  \ Ellipsenbogen zeichnen (hohl oder gefuellt)
18  : _arc  { x y w h b a f }  ." arc "
19          $AA0B 2>buf  x 2>buf  y 2>buf  w 2>buf  h 2>buf
20                       b 2>buf  a 2>buf  f 2>buf ;
21
22  \ Tortenstueck zeichnen (hohl oder gefuellt)
```

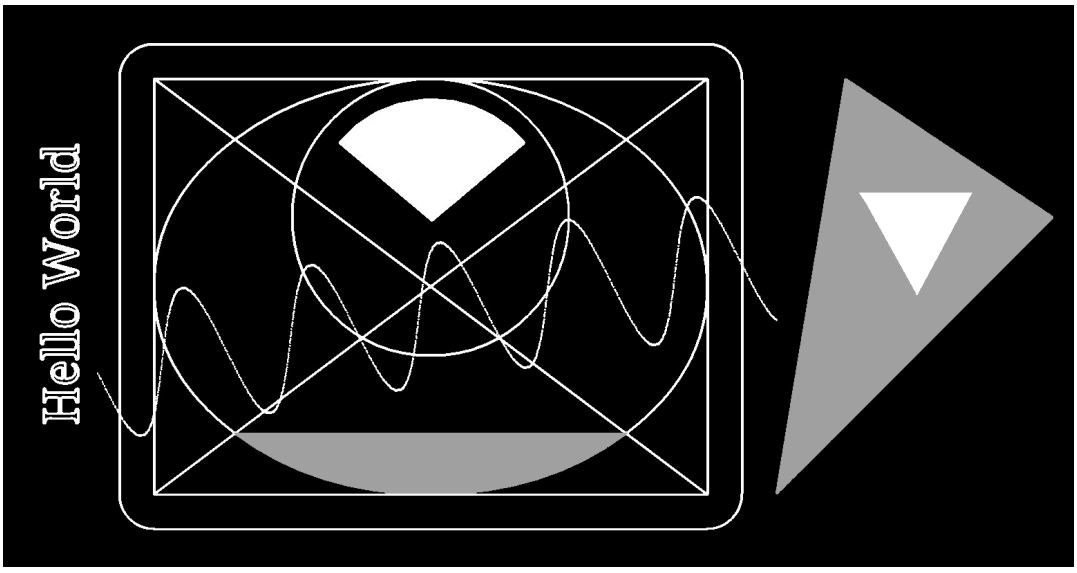


Abbildung 1: Kreis- und Ellipsensegmente, runde Ecken, Dreiecke, Schrift, Drehungen...

```
23  : _pie  { x y w h b a f }  ." arc "
24          $AA0C 2>buf  x 2>buf  y 2>buf  w 2>buf  h 2>buf
25                       b 2>buf  a 2>buf  f 2>buf ;
26
27  \ Zeichensatz und Hoehe waehlen
28  : _font  { typ h }  ." font "
29          $AA0D 2>buf  typ 2>buf  h 2>buf ;
30
31  \ Text einfuegen (big endian)
32  : _text-on   { x y }  ." text-on "
33          $AA0E 2>buf  x 2>buf  y 2>buf ;
34  : _text      { abc }  ." text "  abc 2>buf ;
35  : _text-off  ( -- )   ." text-off "  $AA00 2>buf ;
36
37  \ Bezugsort waehlen
38  : _origin  { x y }  ." origin "
39          $AA0F 2>buf  x 2>buf  y 2>buf ;
40
41  \ Nachfolgenden Plot drehen
42  : _rotate  { a }  ." rotate "
43          $AA10 2>buf  a 2>buf ;
44
45  \ Rechteck mit runden Ecken zeichnen (hohl oder gefuellt)
46  : _rndrect  { x y w h r f }  ." rndrect "
47          $AA11 2>buf  x 2>buf  y 2>buf  w 2>buf  h 2>buf
48                       r 2>buf  f 2>buf ;
49
50  \ Dreieck zeichnen (hohl oder gefuellt)
51  : _triang  { x1 y1 x2 y2 x3 y3 f }  ." triang "
```

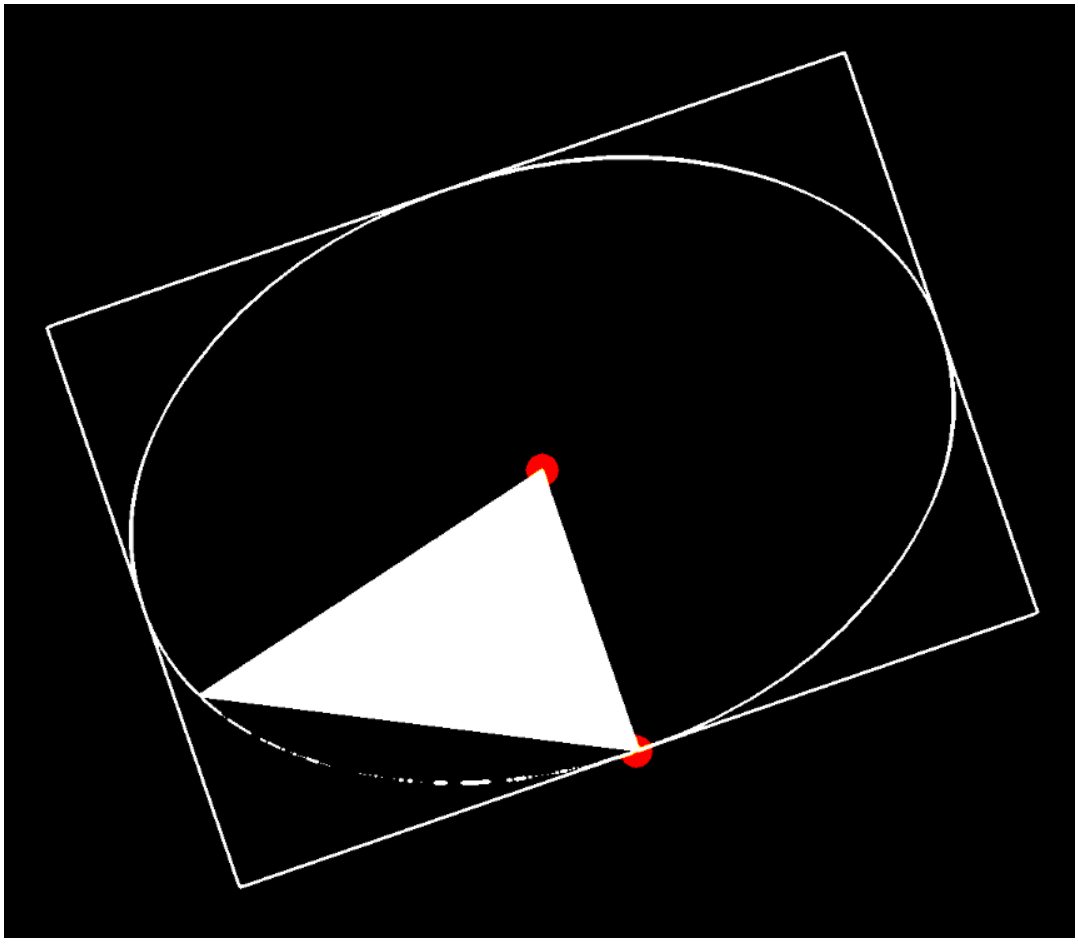


Abbildung 2: Ein schwarzer Bogen hat das “Tortenstück” zum Dreieck werden lassen.

```
52          $AA12 2>buf  x1 2>buf  y1 2>buf  x2 2>buf  y2 2>buf
53                       x3 2>buf  y3 2>buf  f 2>buf ;
54
55  \ Grafikausrichtung speichern
56  : _save  ( -- )  ." save "
57          $AA13 2>buf ;
58
59  \ Grafikausrichtung wiederherstellen
60  : _restore  ( -- )  ." restore "
61          $AA14 2>buf ;
62
63
64  \ ########################
65  \ Eine Grafik formulieren
66  \ ########################
67
68    create-wfile                    \ Ausgabe-Datei erzeugen
69
70  : go                                  _begin      \ Dateibeginn
71
72       0    0    0  255                _color      \ schwarz
73                                       _clear      \ ausfuellen
74    255  255  255  255                 _color      \ weiss
75
76                                       _show       \ zeigen
77                                    1  _pause      \ 1 Sekunde
78
79    400  300  800  600              0  _rect       \ Rechteck
80    400  300 1200  900                 _line       \ Diagonale
81   1200  300  400  900                 _line       \ Diagonale
82
83                                       _show       \ zeigen
84                                    1  _pause      \ 1 Sekunde
85
86    400  300  800  600              0  _oval       \ Ellipse
87
88                                       _show       \ zeigen
89                                    1  _pause      \ 1 Sekunde
90
91    800  500  200                   0  _circle     \ Kreis
92
93                                       _show       \ zeigen
```

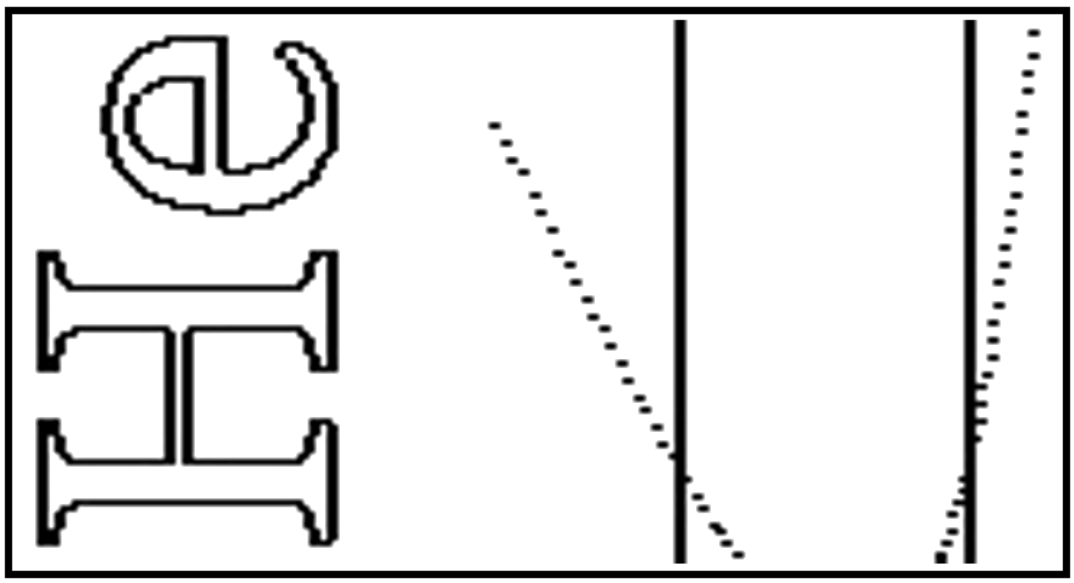


Abbildung 3: Zur Demonstration der Auflösung (invertiert, um Druckerschwärze zu sparen)

```
94                                        1   _pause      \ 1 Sekunde
95
96     627  330  350  350  220  100     1   _pie        \ Torte
97
98     100  100  100  255                    _color      \ grau
99
100    400  300  800  600   45   90     1   _arc        \ Bogen
101
102                                           _show       \ zeigen
103                                       1   _pause      \ 1 Sekunde
104
105    255  255  255  255                    _color      \ weiss
106
107    350  250  900  700   50          0   _rndrect    \ runde Ecken
108
109                                           _show       \ zeigen
110                                       1   _pause      \ 1 Sekunde
111
112                                           _save       \ Drehrichtung
113                                     -10   _rotate     \ Drehung links
114
115                                           _plot-on
116    980 0 do i 200 + i s>f 30e f/ fsin 100e f* 770e f+ f>s
117                                           _plot       \ Sinus
118          loop                             _plot-off
119
120                                           _restore    \ Drehung vergessen
121
122                                           _show       \ zeigen
123                                       1   _pause      \ 1 Sekunde
124
125    100  100  100  255                    _color      \ grau
126
127   1300  900 1700  500 1400  300     1   _triang     \ volles Dreieck
128
129                                           _show       \ zeigen
130                                       1   _pause      \ 1 Sekunde
131
132    255  255  255  255                    _color      \ weiss
133
134   1500  650 1600  460 1400  460     1   _triang     \ volles Dreieck
135
136                                           _show       \ zeigen
137                                       1   _pause      \ 1 Sekunde
138
139    190  940                              _origin     \ neuer Bezugspunkt
140
141                                     -90   _rotate     \ Drehung links
142
143                                   3   70  _font       \ 3 = Serif
144
145    135  100                              _text-on    \ Text x y
146         256 [char] H *  [char] e  +       _text       \ "Hello World "
147         256 [char] l *  [char] l  +       _text
148         256 [char] o *    bl      +       _text
149         256 [char] W *  [char] o  +       _text
150         256 [char] r *  [char] l  +       _text
151         256 [char] d *    bl      +       _text
152                                           _text-off
153
154                                           _show       \ zeigen
155                                     100   _pause      \ 100 Sekunden
156
157                                           _end ;      \ Dateiende
158
159         go      \ Autostart
```

Temperatur–Logger für viele Messstellen

Rafael Deliano

Wenn man viele Temperatur–Messstellen benötigt, und mit den Einschränkungen eines Sensors im TO92–Gehäuse leben kann, bietet sich der Dallas–DS18B20 an. Zwar wäre es möglich, mehrere dieser Sensoren an einem Kabel als Bus zu betreiben, das habe ich jedoch prinzipiell nicht berücksichtigt. Denn für den raschen experimentellen Aufbau ist der Aufwand für Programmierung und Tests auf Kabelbrüche und Kurzschlüsse zu unattraktiv. Hier ist parallele Beschaltung angemessener. Ausgehend von einer experimentellen Platine basierend auf meinem nanoFORTH für den MC68HC908GP32 werden einige praktische Vorschläge zum Einsatz des Sensors gemacht. Und anhand eines Messversuchs an einem optischen Durchflusssensor eine Anwendung kurz dargestellt.

Aufbau

Im Prinzip denkbar einfach. Es wird jeweils nur ein Portpin mit pullup–Widerstand **Abb. 1** je Sensor benötigt. Man kann deshalb am GP32–Controller problemlos etwa 24 Kanäle verteilt über Ports A, B, C, D bereitstellen **Abb. 2**.

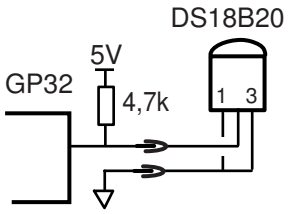


Abbildung 1: Schaltung

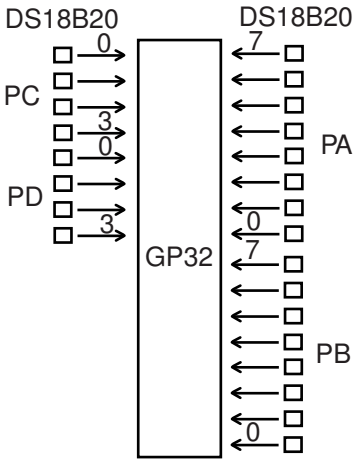


Abbildung 2: Blockschaltbild

Die Abtastrate ist ca. 1sec. Der GP32 kann die Messwerte sofort direkt über V24 an den PC weitergeben. Wenn das nicht möglich ist, wird lokaler Speicher benötigt. Hier wurde alternativ ein 24C512–EEPROM in DIL8 oder eine Adapterplatine, die SRAM bzw. SD–Karte enthalten kann, vorgesehen **Abb. 3**.

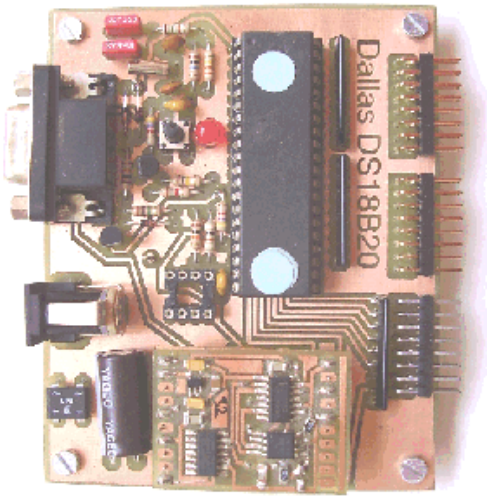


Abbildung 3: Leiterplatte

An der Leiterplatte hat jeder Sensor einen 2pol–Stecker. Normalerweise befinden sich die ICs im idle–Modus und der Pegel ist 5V. Wenn eines verpolt gesteckt wurde, wird durch die interne Diode die Spannung auf 0,6V geklemmt und ist als low–Pegel erkennbar. Eine Test–Routine kann also diese Verdrahtungsfehler sofort erkennen. Eleganter ist es, Drähte unterschiedlicher Farbe zu verwenden, um Falschpolung sichtbar zu machen **Abb. 4**. Die Zuordnung vom Kanal zum Messpunkt nach Montage erfolgt bei Aufbauten mit vielen Sensoren am einfachsten durch Antippen des Sensors mit dem Lötkolben, während der Logger zyklisch den Datensatz mit den 24 Sensoren auf dem Bildschirm ausgibt.

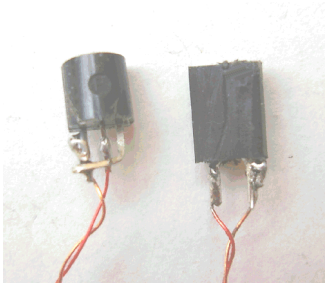


Abbildung 4: Sensor, Buchse

Software

Erforderlich sind Routinen in Assembler, die exaktes Timing sicherstellen **Abb. 5**. Sie sind abhängig vom Controller und dessen Takt. In **Listing 1** deshalb nur auszugsweise dargestellt. Wenn Interrupts verwendet werden, müssen diese derweil gesperrt sein.

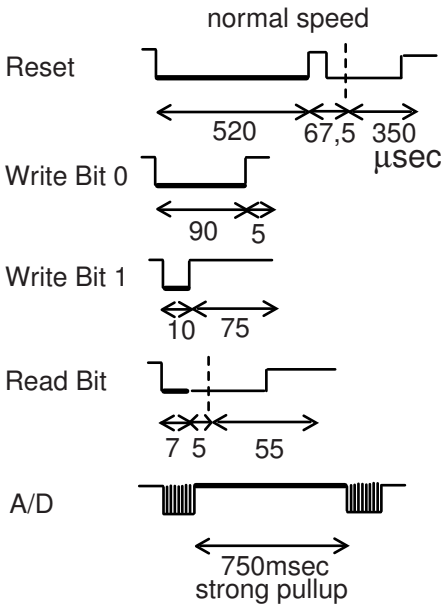


Abbildung 5: Timing

Der Portpin ist normalerweise als Input geschaltet. Oder, im Schaltbild mit breiter Linie dargestellt **Abb. 5**, als Output low bzw. als "strong pullup" Output high. Oder er muss gelesen werden. Damit man die nötigen Assembler–Bitbefehle mit der Nummer des Sensors `S# = 0 ... 23d` adressieren kann, sind sie im Listing 1 in Sprungziel–Tabellen `P0!... P@` angelegt. Jeweils mit NOP–Opcodes auf 5 Bytes aufgepolstert, so dass ein einheitlicher Offset im X–Register verwendet werden kann.

Die Routinen zum Lesen des ROMs will man mit Ausdruck der Daten für Debugging (`SN"`), aber für Normalbetrieb dann ohne Ausdruck (`SN`). Gleiches gilt für das Auslesen des Scratchpads, über das die Daten zurückkommen. Hier so ausgeführt, dass, wenn ein Sensor z.B. durch Kabelbruch oder Kurzschluss defekt ist, der Meswert „000 " geliefert wird. Es wurde nur die default–Einstellung 12Bit mit langer Messzeit 750msec und Auflösung 0,0625°C/Bit implementiert. Wenn man Zahlenkolonnen loggt, kann man die Rohdaten dezimal ausdrucken und die Skalierung *0,0625 anschließend z.B. in Mathcad machen. Fürs Debugging ist aber Ausgabe als xxx,xx °C direkt im Terminalprogramm angenehmer, die passende Routine für positive Temperaturen ist in **Listing 2** dargestellt.

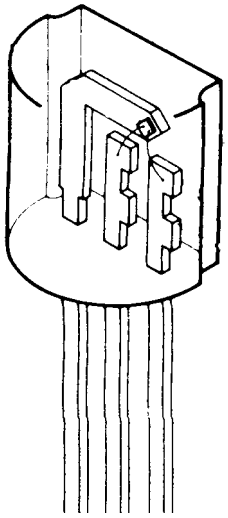


Abbildung 6: Interner Aufbau TO92–Gehäuse

Sensor

Der Wärmewiderstand vom Chip zur Umgebungstemperatur teilt sich beim TO92–Gehäuse **Abb. 6** typisch 50:50 auf das Kunstoffgehäuse und die Anschlusspins auf. Man kann die Stirnseite mit Sekundenkleber auf die Messstelle kleben. Da das Bauteil nur etwa 1 EUR kostet, ist eine Wiederverwendung verzichtbar. Um Wärmeabfuhr über die Pins dann zu verringern, schneidet man die Beinchen auf 3mm ab und führt die Kabel als dünnen Kupferlackdraht aus **Abb. 4**. Die Drähte müssen nicht einmal sonderlich verdrillt werden, da die Übertragung über den onewire–Bus digital recht störsicher erfolgt.

Alternativ kann man sich auch auf die Pins konzentrieren. Für die Messung von Wassertemperatur **Abb. 7** wurden die Pins nicht gekürzt, sondern jeweils auf zwei Kupferbleche gelötet, und dann alles schwarz lackiert, um für elektrische Isolierung zu sorgen. Gleiches Vorgehen, um die Oberflächentemperatur eines Teflonrohrs zu messen **Abb. 8**. Diesmal allerdings nur Kupferblech für die GND–Pins.

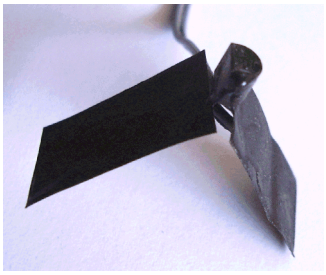


Abbildung 7: Flüssigkeit

Abbildung 8: Kupferfolie auf Rohr

Optimierungen

Bei 24 Sensoren dauert eine komplette Messung derzeit, weil sie völlig sequentiell erfolgt, ca. 24 sec, was arg lang ist. Man kann das IC zwar auf weniger als 750 msec umprogrammieren, verliert dadurch aber entsprechend Auflösung. Denkbar wäre die Umstellung der Software auf eine mehr gleichzeitige Messung **Abb. 9**. Und parallel dazu kann man auch die Ausgabe auf dem PC vornehmen, die bei 9600 Baud etwa 10 msec/Sensor benötigt.

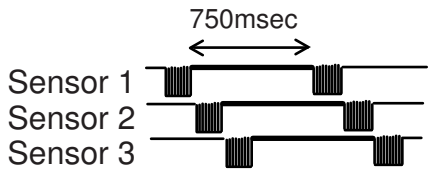


Abbildung 9: Verschachtelte Messzeit

Anwendung Test Durchflussmessung

Es gibt von Optek Gabellichtschranken, die man auf einen durchsichtigen Kunstoffschlauch aufstecken kann **Abb. 10**. Damit kann man auch „durchsichtige“ Flüssigkeiten wie Wasser erkennen. Allerdings ist das Signal klein, die Bauteilstreuung muss vom Controller mit A/D–Wandler und Korrekturtabelle im EEPROM kompensiert werden.

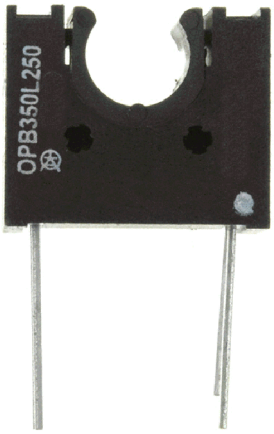


Abbildung 10: OPB350L250 „liquid sensor“

Ein kurzer Test mit der Grundschaltung **Abb. 14** war vielversprechend. Aber die Temperaturdrift scheint kritisch, auch das Datenblatt weist auf diesen Punkt hin **Abb. 16**.

Um das zu überprüfen, wurde eine simple Mechanik **Abb. 11** mit drei Temperatursensoren aufgebaut. Damit kann man Signalsprünge Wasser/Luft erzeugen, um Zeitkonstanten zu bestimmen. Am Sensor **Abb. 12** wurden die beiden GND–Pins des DS18B20 mit den GND–Pins des Optokopplers im Layout über eine gemeinsame Massefläche thermisch verbunden. Dessen Signal wird mit dem 8–Bit–A/D–Wandler des GP32 aufgezeichnet. Die drei Temperaturwerte wurden gefolgt vom Wert des A/D–Wandlers ausgedruckt **Abb. 13** und vom Terminalprogramm mitgeschrieben. Das Textfile kann man dann in Mathcad einlesen, um Grafiken zu machen **Abb. 15**.

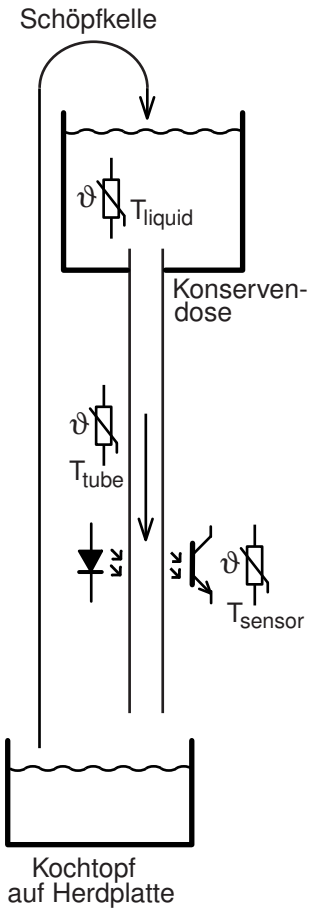


Abbildung 11: Anordnung Temperaturtest

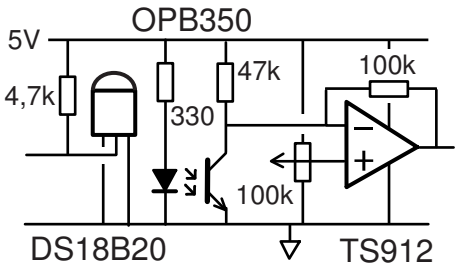


Abbildung 12: Schaltung Temperaturtest

Die Daten zeigen, dass das Signal bei hoher Temperatur zwar schrumpft, aber die Pegel absehbar gut unterscheidbar sind, wenn man die Schaltschwelle anhand des Temperaturfühlers nachführt. Die Abtastrate ist mit 2,5sec niedrig, aber man sieht, dass sie für die Zeitkonstanten thermischer Signale ausreicht.

```
13.50   18.00   20.25   80
13.50   18.06   20.25   80
13.50   18.06   20.25   80
13.56   18.06   20.25   80
13.56   18.12   20.25   80
13.56   18.12   20.25   80
13.56   18.18   20.25   80
13.56   18.18   20.25   80
13.56   18.18   20.31   80
13.62   18.18   20.31   80
13.62   18.25   20.31   80
13.62   18.25   20.25   80
13.62   18.31   20.31   80
13.62   18.31   20.31   238
14.12   18.31   20.25   238
15.12   18.31   20.25   238
```

Abbildung 13: Tabelle: Anfang des Logfiles mit Messergebnissen zu Abb. 15

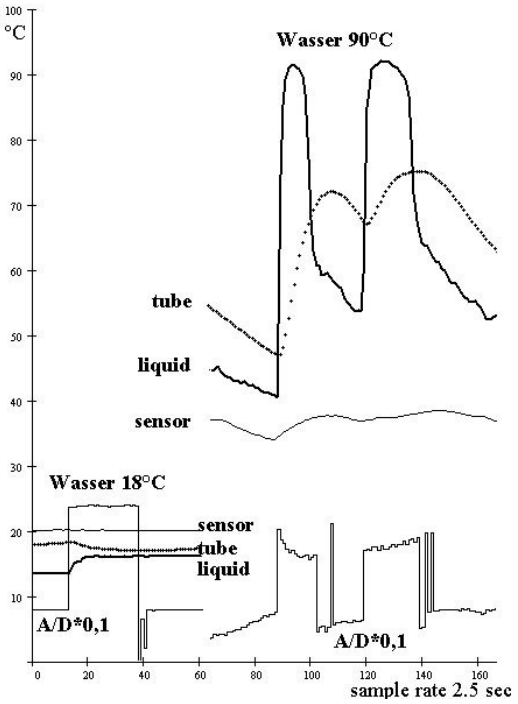


Abbildung 15: Messergebnisse

Einsatz in Serie

Das Rohr darf sich nicht durch Alterung verfärben oder Haftung von Rückständen auf der Innenwand begünstigen. Teflon scheint da gute Wahl zu sein, ist auch sehr temperaturbeständig. Um Bauteilstreuung zu kompensieren, muss die Schaltschwelle für jeden Sensor im eingebauten Zustand bequem kalibrierbar sein. Das LED sollte gepulst betrieben werden, um Alterung zu minimieren. Zudem kann man dann auch Umgebungslicht kompensieren. Nicht jede Mechanik ist mit sinnvollem Aufwand völlig lichtdicht baubar. Und der Schlauch kann eine parasitäre lightpipe bilden.

Die Kompensation des Temperaturgangs durch einen Thermistor scheint jedenfalls praktikabel. Für Wasser ist nur der Bereich von 0 ... 100°C von Bedeutung, und da ist der Verlauf linear.

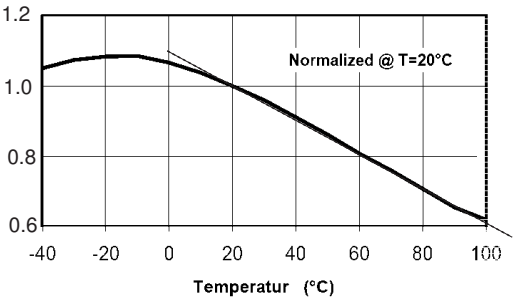


Abbildung 16: Temperaturabhängigkeit laut Datenblatt

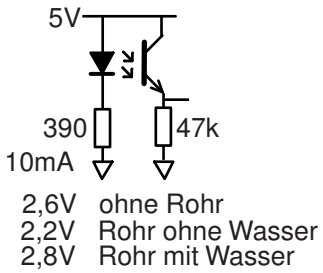


Abbildung 14: Simple Testschaltung

Links

- nanoFORTH `http://www.embeddedforth.de/`
- DS18S20 1–Wire Parasite–Power Digital Thermometer `http://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18S20.html`

Listing 1: Treiber DS18B20

```
\ SRC0.txt

\ 68HC908 @ 2,45MHz

1 ZVARIABLE >S     2 ZVARIABLE A^

TABLE P0!   \ portpin output 0
[CODE
PA 7 MBC, DPA 7 MBS, RTS,
PA 6 MBC, DPA 6 MBS, RTS,
PA 5 MBC, DPA 5 MBS, RTS,
...

TABLE P1!   \ portpin output 1
[CODE
PA 7 MBS, DPA 7 MBS, RTS,
PA 6 MBS, DPA 6 MBS, RTS,
PA 5 MBS, DPA 5 MBS, RTS,
...

TABLE PT!   \ portpin tristate / in
[CODE
DPA 7 MBC, RTS, NOP, NOP,
DPA 6 MBC, RTS, NOP, NOP,
DPA 5 MBC, RTS, NOP, NOP,
...

TABLE P@    \ read portpin to Carry
[CODE
 1 $ PA 7 BBS,  1 $: RTS, NOP,
 2 $ PA 6 BBS,  2 $: RTS, NOP,
...
 8 $ PA 0 BBS,  8 $: RTS, NOP, SOLVE$

:CODE RESET-TM    \ (  - F )  F = 1 : ok
XSAVE           STX, \ save X-register
      >S       LDX,  \ start lowpuls 480usec
P0! ,X          JSR,
      D% 190 #. LDA, \ 240usec
1 $: 1 $   A.   DBN,
      D% 190 #. LDA, \ 240usec
2 $: 2 $   A.   DBN,
PT! ,X          JSR, \ delay high 70usec
      D% 52  #. LDA,
3 $: 3 $   A.   DBN, \ sample DIN
      A.        CLR,
P@ ,X           JSR,
4 $             BCS,
      A.        DEC,
4 $: XSAVE     LDX, \ restore X-register
               DEX, \ put flag on stack
            0,X STA,
                DEX,
            0,X STA, \ delay 410usec
      D% 163 #. LDA, \ 205usec
5 $: 5 $   A.   DBN,
      D% 163 #. LDA, \ 205usec
6 $: 6 $   A.   DBN,
                RTS,
CODE;

:CODE TM-B1!    \ (  -  )
... CODE;

:CODE TM-B0!    \ ( - )
... CODE;

: TM-C!      \ ( C1 -  )
7 0 DO DUP 1 AND IF TM-B1!
ELSE TM-B0! THEN 1SHIFT> LOOP DROP ;

:CODE (TM-C@) \ (  - C1 )
... CODE;

:CODE 44-TM-C!  \ (  -  ) Convert-T command
, TM-B0! JSR, , TM-B0! JSR, , TM-B1! JSR,
, TM-B0! JSR, , TM-B0! JSR, , TM-B0! JSR,
, TM-B1! JSR, , TM-B0! JSR,
XSAVE           STX,
      >S        LDX,
P1! ,X          JSR, \ STRONG-ON
XSAVE           LDX, RTS, CODE;

:CODE STRONG-OFF        \ ( >S -  )
XSAVE           STX,
      >S        LDX,
PT! ,X          JSR,
XSAVE           LDX, RTS, CODE;

8C CONSTANT POLY    \ TM reverse

: CRC+              \ ( CRC C1 - CRC' C1 )
SWAP OVER
XOR 7 0 DO DUP 0001 AND IF 1SHIFT> POLY
XOR ELSE 1SHIFT> THEN LOOP SWAP ;

: TM-C@                    \ (  - C1 )
0 7 0 DO (TM-C@) IF 100 OR THEN 1SHIFT>
LOOP FF AND ;

: SN"      \ (  -  )  Serial number, print
33 TM-C! \ Read-ROM Command
00  \ init CRC
TM-C@ CRC+ CR CH. SPACE ." FamilyCode"
TM-C@ CRC+ CR CH.   \ CH. : print byte hex
TM-C@ CRC+ CH. TM-C@ CRC+ CH. TM-C@ CRC+ CH.
TM-C@ CRC+ CH. TM-C@ CRC+ CH.
SPACE ." Serial Number"
TM-C@ CR CH. CH. ." CRC" CR ;

: SN      \ (  -  )  Serial number, silent
33 TM-C! \ Read-ROM Command
00      \ init CRC
TM-C@ CRC+ DROP TM-C@ CRC+ DROP
TM-C@ CRC+ DROP TM-C@ CRC+ DROP
TM-C@ CRC+ DROP TM-C@ CRC+ DROP
TM-C@ CRC+ DROP TM-C@ = LNOT IF 0 A^ ! THEN ;

: TEMP      \ (  -  ) temperature conversion
44-TM-C!  D% 750 MSEC   STRONG-OFF ;

: SP"      \ (  -  ) read scratchpad, print
BE TM-C!   \ Read Scratchpad Command
00         \ init CRC
TM-C@ CRC+  CH.  TM-C@ CRC+  CH.  TM-C@ CRC+  CH.
TM-C@ CRC+  CH.  TM-C@ CRC+  CH.  TM-C@ CRC+  CH.
TM-C@ CRC+  CH.  TM-C@ CRC+  CH.  ." Scratch Pad"
TM-C@ CR CH. CH. ." CRC" ;

: SP     \ (  -  ) read scratchpad, silent
BE TM-C! \ Read Scratchpad Command
00      \ init CRC
TM-C@ CRC+ A^ 1+ C! TM-C@ CRC+ A^    C!
TM-C@ CRC+ DROP  TM-C@ CRC+ DROP
TM-C@ CRC+ DROP  TM-C@ CRC+ DROP
TM-C@ CRC+ DROP  TM-C@ CRC+ DROP
TM-C@ = LNOT IF 0 A^ ! THEN ;

: >S!    \ ( S# -  )
5 U* DROP >S C! ;

: S      \ ( S# - UN1 )
         \ S# = 0 ... 23d number of sensor
>S!  0 A^ !
RESET-TM IF SN TEMP THEN
RESET-TM IF SN SP THEN A^ @ ;

CR ." -> SRC1.txt"  CR
```

Listing 2: Druckroutine

```
\ SRC1

\ Numeric printing 0 ... 256 Celsius
\ Dallas Sensor: -55 to +125 Celsius

: CD.     \ ( C  -  ) C = 0 - 63h
\ emits 0 - 99d , rightadjusted,
\ decimal, 2 chars without leading SPACE
DUP 9 U> IF
  9 1 DO
  0A - DUP
  0A U< IF LEAVE I THEN
  LOOP 30 + EMIT
ELSE SPACE
THEN 30 + EMIT ;

: 0CD.     \ ( C  -  ) C = 0 - 63h
\ Emits 00 - 99d , rightadjusted,
\ decimal, 2 chars with leading SPACE
DUP 9 U> IF
  9 1 DO
  0A - DUP
  0A U< IF LEAVE I THEN
  LOOP 30 + EMIT       \ first letter
ELSE 30 EMIT
THEN 30 + EMIT ;

: 3CD.    \ ( C -  ) C = 0 - FF
\ emits 0 - 255d, rightadjusted, decimal, 3 chars
\ without leading SPACE
DUP 63 U>
IF DUP C7 U> IF C8 - 32 ELSE 64 - 31 THEN EMIT 0CD. ELSE SPACE CD. THEN ;

: 256,00.              \ ( UN1 -  )
DUP D% 25601 U< IF 0 D% 100 U/MOD 3CD. ." ," 0CD. ELSE DROP ." overflow "  THEN ;

: TEMP.        \ ( UN1 -  ) print
D% 100 U* D% 16 U/ 256,00. SPACE ;

CR ." ->SRC2"  CR
```

Forth als Prozess 1 unter Linux

Carsten Strotmann

Nicht jedes (embedded) System benötigt ein vollständiges modernes Betriebssystem, jedoch ist es bequem, auf die Treiber eines vollen Betriebssystems zurückgreifen zu können, anstatt alle Gerätetreiber in Forth zu programmieren. Dieser Artikel zeigt, wie ein Forth–System direkt auf dem Linux-Kern ausgeführt werden kann, ohne den Ballast eines vollen Linux-Betriebssystems.

Gforth "almost native"

Moderne Betriebssysteme umfassen heute mehrere Gigabyte an Programmen und Daten. Die Systeme sind komplex und es ist aufwendig, das System manuell aktuell zu halten (Sicherheitsupdates). Möchte der Entwickler Linux einsetzen, um bei einem "embedded"–System unabhängig von der Hardware zu sein, so ist das Betriebssystem ein Ballast. Ziel ist ein sehr simples System, welches aus nur zwei Komponenten besteht: dem Linux-Kern und dem Forth-System.

Linux Bootprozess

Ausgangspunkt meiner Versuche ist das "gforth almost native" von Anton Ertl[1]. Um das Konzept von "gforth almost native" zu verstehen, braucht es ein wenig Hintergrundwissen über den Linux-Bootprozess:

1. Firmware/BIOS/UEFI
2. Bootloader (Grub, syslinux, lilo ...)
3. Linux–Kernel (vmlinux)
4. Init–System (SYSV-Init, systemd, upstart, OpenRC ...)
5. Netzwerk, Hintergrunddienste, grafische Oberfläche
6. Anwenderprogramme (gforth)

Im Schritt 4 wird gestartet, was der Benutzer als Betriebssystem erkennt. Anstatt eines der von Linux benutzten Init-Systeme kann aber ein beliebiges Programm gestartet werden. Also auch ein Forth. In meinen Versuchen habe ich mit gforth[3] gearbeitet, andere unter Linux ausführbare Forth–Systeme sollten ähnlich funktionieren.

Der Bootprozess unfasst dann die folgenden Schritte:

1. Firmware/BIOS/UEFI
2. Bootloader (Grub, syslinux, lilo ...)
3. Linux–Kernel (vmlinux)
4. Forth

Auf der Festplatte (oder USB-Stick, SD-Karte) befinden sich im Minimalfall nur noch zwei "Programm-Dateien": der Linux–Kern und das Forth–Programm. Übersichtlich.

Statisch gelinktes Gforth

In einem "normalen" Linux-System sind die meisten Programme dynamisch gelinkt, benötigte Bibliotheken werden beim Start der Programme aus den dynamischen Link-Libraries (ähnlich der DLLs unter Windows) hinzugeladen. Das ist in einem Linux-System mit vielen Prozessen sinnvoll, da der Programmcode für die Libraries (z.B. für die C-Libarary) nur einmal gespeichert wird. In unserer Anwendung ist dynamisches Linken aber überflüssiger Ballast, es macht das System komplexer.

Linux-Programme können statisch gelinkt werden, dabei werden alle Programmbestandteile inklusiv der Bibliotheken in die Programmdatei geschrieben. Felix von Leitner (Fefe) hat mit der Diet–Libc[9] eine kompakte C–Bibliothek geschrieben, welche für das Linken von kleinen statischen Linux–Programmen optimiert ist. Die Schritte, um gforth mit der Diet–Libc neu zu übersetzen, sind im Wiki der Forth–Gesellschaft beschrieben[10].

Monolithischer Kernel

Ähnlich ist es beim Linux–Kernel. Die Linux–Kernel der Distributionen sind modular aufgebaut, es müssen verschiedenste Rechnerarchitekturen und Geräte unterstützt werden. Gerätetreiber sind in Kernelmodule ausgelagert, der Linux–Kern prüft die Hardware und lädt die Kernelmodule bei Bedarf. Ein modularer Kernel ist flexibel, aber auch kompliziert. Wenn wir die Hardware kennen, auf der Linux und Forth eingesetzt werden soll, so kann der Linux–Kernel passend auf die Hardware angepasst werden (nur die benötigten Gerätetreiber werden in den Kernel eingebettet). Der Kernel kann dann "monolithisch" kompiliert werden, d.h. alle benötigten Treiber sind schon Bestandteil der Kernel–Datei.

Zum Erstellen eines "monolithischen" Linux–Kerns startet man ein volles Linux-System (z.B. ein Knoppix–Live–Linux[12] von DVD oder CD-ROM) und lädt dann den Linux-Quellcode[11]. Den Kernel–Quellcode nach `/usr/src/linux` entpacken und als Superuser `root` in das Verzeichnis wechseln. Dort den Befehl `make localmodconfig` eingeben. Dieser Befehl untersucht das aktuell gestartete Linux–System, und konfiguriert den Quellcode basierend auf dem aktuell gestarteten Linux (Gerätetreiber werden nur aktiviert, wenn die Treiber im aktuellen System erfolgreich geladen wurden). Hierdurch wird eine Linux–Kernel–Konfiguration erstellt, welche ideal auf die eigene Hardware abgestimmt ist.

Die Linux–Kernel–Konfiguration ist nun aber noch modular. Mit dem Befehl `make menuconfig` wird das Linux–Kernel–Quellcode–Konfigurationsmenue aufgerufen, dort "Enable loadable module support" abwählen, um einen modularen Kernel zu bekommen, siehe Abbildung 1.

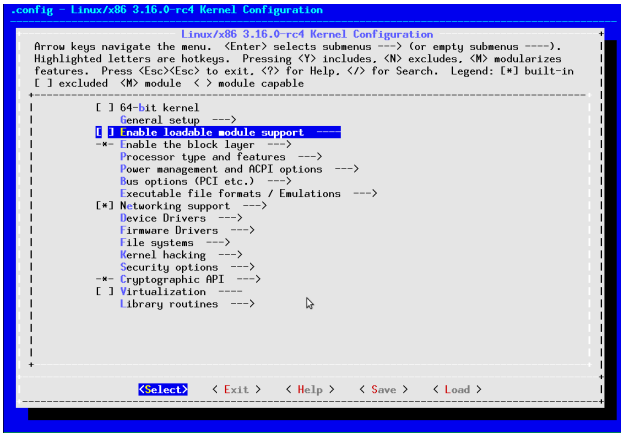


Abbildung 1: Module in der Linux–Kernel–Konfiguration ausschalten

Mit dem Befehl `make bzImage` wird der neue Linux–Kernel gebaut. Die Kernel-Datei (hier für i386 Intel PC) findet sich unter `/usr/src/linux/arch/x86/boot/bzImage` und kann von dort manuell an das gewünschte Ziel kopiert werden (z.B. in die Root–Partition `/`).

Das System zusammenstellen

Im Folgenden beschreibe ich meine Beispiel–Installation. Die Pfade können auch anders gewählt werden, wobei zu beachten ist, dass der Linux–Kernel nur das Root-Dateisystem "/" anhängt (mounted), so dass alle Dateien auf der Root-Partition liegen müssen.

Von Gforth habe ich die Dateien `gforth` und die Image-Datei `gforth.fi` in das Verzeichnis `/gforth` kopiert. In diesem Verzeichnis speichere ich auch den Forth-Quellcode. Die Gforth–"Autostart–Datei" `.gforthrc` (dieser Forth-Code wird beim Start von Gforth automatisch geladen) liegt im Hauptverzeichnis `/.gforthrc`. Diese Datei lädt bei mir u.A. die Syscall–Wörter (siehe unten).

Der Linux–Kernel (in der Datei mit dem Namen "bzImage") liegt unter `/vmlinuz`. Als Bootloader benutze ich Grub2. Listing 1 zeigt einen Beispiel–Menueeintrag für Gforth. `root=/dev/sda3` definiert die Partition des Root–Dateisystems, welches Linux automatisch einhängen soll. `rw` spezifiziert, dass das Dateisystem zum Lesen und Schreiben (Read–Write) eingehangen wird. `init=/gforth/gforth` gibt an, welches Programm der Linux–Kern als Prozess 1 starten soll, in diesem Fall Gforth. `quiet` unterdrückt die Meldungen des Linux-Kernels und sorgt für einen schnellen Start. Bei Problemen kann dieser Eintrag herausgenommen werden und mögliche Fehlermeldungen erscheinen auf dem Bildschirm.

Mit dieser Konfiguration startet Gforth in ein paar Sekunden. Schneller als bei alten MS–DOS Systemen. Der Forth–Programmierer hat nun die gesamte Maschine zur Verfügung, der Forth–Prozess läuft unter dem Super–User–Account "root" (Vorsicht!). Der Forth-Programmierer hat ungehinderten Zugriff auf die gesamte Hardware, inkl. Festplatten, USB-Sticks, Grafikkarte (Framebuffer) und Netzwerkschnittstellen.

Gforth Verlassen

Der Befehl zum Beenden des Gforth, `bye`, übergibt die Ausführung wieder an das aufrufende Programm. In einem "normalen" Linux–System ist das eine Shell oder die grafische Oberfläche, wurde Gforth aber als Prozess 1 gestartet, so bekommt der Linux-Kernel die Kontrolle zurück. Da das Programm 1 in einem Linux-System sich nie beenden darf, quittiert der Linux-Kernel diese Situation mit einem "Panic"–Fehler.

Das Gforth als Prozess 1 kann nur über einen Reboot oder ein Herunterfahren (Shutdown) des Systems beendet werden. Reboot oder Shutdown werden über sogenannte Syscalls zum Linux-Kernel ausgeführt. Syscalls sind die API des Linux–Kernels, eine Liste der verfügbaren Syscalls findet sich im Link 2.

Die Syscalls werden von Assembler heraus aufgerufen, die Syscall–Nummer wird im (R)AX-Register übergeben, weitere Parameter in den anderen Registern oder auf dem Stack. Das Listing im Anhang zeigt beispielhaft Syscall-Wort-Definitionen für Reboot, Poweroff, Get-PID (Prozess–ID ermitteln) und Sync (Festplatten-Buffer schreiben). ABI-CODE[8] sorgen für Quellcode, der zwischen verschiedenen Gforth–Versionen auf der gleichen Architektur portabel bleibt. Bücher über x86–Assembler finden sich im Internet als eBook, einige sogar kostenlos[5][6][7]. Besonders gut hat mir das Buch "Introduction to Computer Organization" von Robert G. Plantz gefallen[4]. Die Syscall-Beispiele im Anhang sind für 32Bit–x86–Linux, Versionen für 64Bit–Linux (amd64), x32 (32Bit–Linux auf 64Bit–x86–Prozessoren) und ARM finden sich im Wiki[10].

GNU/Forth/Linux

Aus einem Linux–Kernel und einem Forth–System lässt sich ein kompaktes und schnelles Forth–Computer-System bauen. Wer die Erstellung eines eigenen monolithischen Kernels und eines statisch–gelinkten Gforth scheut, findet im Wiki [10] eine Imagedatei für einen bootbaren USB-Stick nebst Anleitung. Ein vorbereiteter USB-Stick kann für 10 Euro (inkl. Porto) beim Mikrokontrollerverleih, Email `mailto:mcv@forth-ev.de` bestellt werden.

Viel Spaß mit GNU/Forth/Linux!

Literatur und Links

1. Anton Ertl: Almost native gForth http://www.complang.tuwien.ac.at/forth/gforth/almost-native/
2. Linux Syscall Table http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html
3. GNU Forth (gforth) http://www.gnu.org/s/gforth/
4. Introduction to Computer Organization von Robert G. Plantz, PDF eBook bei Lulu http://www.lulu.com/shop/robert-g-plantz/introduction-to-computer-organization-ebook/ebook/product-21369110.html
5. PC Assembly Language von Paul Carter, Kostenloses PDF eBook bei Lulu http://www.lulu.com/shop/paul-carter/pc-assembly-language/ebook/product-17380062.html
6. Art of Assembly Language, 2nd Edition von Randall Hyde http://www.nostarch.com/assembly2.htm
7. Wikibooks X86 Assembly http://en.wikibooks.org/wiki/X86_Assembly
8. A. Ertl u. D. Kühling – ABI-CODE: Increasing the portability of assembly language words http://www.complang.tuwien.ac.at/anton/euroforth/ef10/papers/ertl.pdf
9. fefe: diet libc - a libc optimized for small size http://www.fefe.de/dietlibc/
10. Gforth als PID 1 unter Linux in Forth-Gesellschaft Wiki http://forth-ev.de/wiki/doku.php/projects:gforth-as-pid-one:start
11. Linux Kernel Repository http://kernel.org
12. Knoppix-Live-Linux System http://www.knopper.net/knoppix/index.html

Listings

Listing 1: Grub2–Linux–Kernel–Konfiguration unter Debian 7

```
% cat /etc/grub.d/40_custom
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.

menuentry 'GNU/Forth almost native' --class debian --class gnu-linux --class gnu --class os {
        load_video
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='(hd0,msdos1)'
        search --no-floppy --fs-uuid --set=root 3413a572-49e9-47f0-9a27-fb3daa7c13b2
        echo    'Loading GNU/Forth ...'
        linux   /vmlinuz root=/dev/sda3 rw init=/gforth/gforth quiet
}
```

Listing 2: Linux–32Bit–Syscalls

```
1
2   abi-code sys-getpid ( -- pid ) \ get process ID for gforth
3      20  #   ax mov  \ get-pid syscall
4      $80 #      int  \ execute syscall
5      ax      cx mov  \ save result
6      4 sp d) ax mov  \ get stackpointer
7      4 #     ax sub  \ make space for result on stack
8      cx    ax ) mov  \ move result to stack
9                ret  \ return from code
10  end-code
11
12  abi-code sys-reboot ( -- ) \ shutdown and reboot machine
13         88 # ax mov  \ reboot syscall
14  $fee1dead # bx mov  \ magic 1
15   85072278 # cx mov  \ magic 2
16  $01234567 # dx mov  \ command (restart)
17        $80 #    int  \ execute syscall
18                 ret  \ we don't expect to come back
19  end-code
20
21  abi-code sys-poweroff ( -- ) \ shutdown and poweroff machine
22         88 # ax mov  \ reboot syscall
23  $fee1dead # bx mov  \ magic 1
24   85072278 # cx mov  \ magic 2
25  $4321fedc # dx mov  \ command (poweroff)
26        $80 #    int  \ execute syscall
27                 ret  \ we don't expect to come back
28  end-code
29
30  abi-code sys-sync ( -- ) \ commit buffer cache to disk
31    36 #      ax mov  \ sync syscall
32   $80 #         int  \ execute syscall
33    4 sp d)   ax mov  \ return stackpointer
34                 ret  \ return
35  end-code
36
37  \ redefine "bye"
38  : bye sys-sync sys-poweroff ;
```

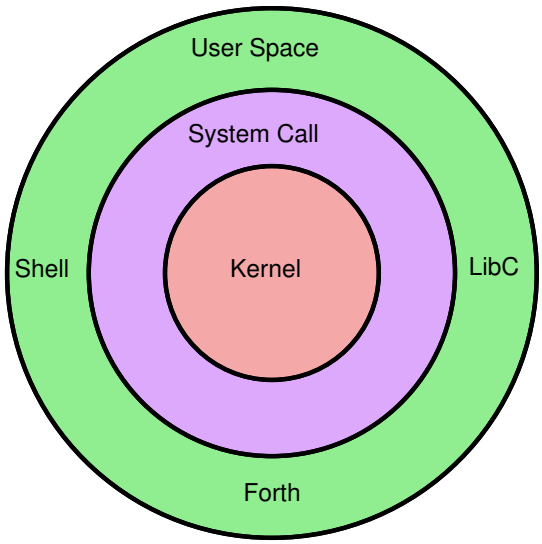


Abbildung 2: Linux–Syscall–Schnittstelle

Ein IP–Stack für das Tiva Connected Launchpad

Bernd Paysan

Auf der Tagung haben wir ein Projekt gestartet, eine modernisierte Version von Heinz Schnitters *Open Network Forth (ONF) auf moderne Mikrocontroller zu bringen. Der erste Schritt dafür ist ein IP–Stack auf so einem Controller, damit wir schon mal Daten zwischen PC und Controller hin- und herschicken können.*

Tiva Connected Launchpad

Das Board, das ich mir für den Start dieser Netzwerkexperimente ausgesucht habe, ist das Tiva Connected Launchpad von Ti. Das kostet etwa 20€, ist also durchaus erschwinglich. Wie bei Ti üblich, steckt man das Board über eine USB–Mini–Buchse an den PC, und bekommt darüber Strom, Flash–Programmer und serielle Schnittstelle. Also, einfach anstöpseln, und man kann mit dem Gerät reden.

Auf dem Board ist ein TM4C1294NCPDT SoC, ein recht gut ausgestatteter Controller. Kern ist ein ARM Cortex M4, der mit 120MHz laufen kann. Es gibt 1MB Flash und 256kB RAM, sowie 6kB EEPROM. Auch die Kommunikationsinterfaces sind reichhaltig, es gibt 8 UARTs, 4 SSI–Module, 10 I²C–Module, 2 mal CAN–Bus, einen Ethernet PHY+MAC (ja, nur die Spulen müssen extern sein) und einen USB 2.0 OTG–Bus. Auch bei den ADCs ist der Kontroller luxuriös ausgestattet: 2 mal 12 Bit mit je 2 Megasamples maximaler Samplerate. Auch die Z280, die Heinz vor fast 30 Jahren angeschafft hat, waren üppig ausgestattet… der TM4C1294NCPDT kostet selbst in 10k–Stückzahlen etwa 10€, ist also auch nicht für den absoluten Low–Cost–Markt gedacht.

Matthias Koch hat sein mecrisp–stellaris [1] schnell auf dem Board zum Laufen gebracht, und schon mal etwas mit dem Ethernet–Treiber angefangen. Wie die kleinere Version vom mecrisp für den MSP430 ist das mecrisp–stellaris ein Native Code Forth, das Maschinencode generiert. Als Optimierungen gibt es Inline und Constant Folding, das lässt sich schön kompakt implementieren.

Als der Ethernet–Treiber von Matthias bei mir ankam, ging das Empfangen von Ethernet–Paketen und Anzeigen als Dump. Das Senden ging nicht, obwohl es nach Doku eigentlich korrekt implementiert war. Das hat sich dann als Bug im Chip herausgestellt: Mit den neueren „komplexen Descriptoren“ geht es, mit den alten einfachen Deskriptoren halt nicht… auch bei Ti kann man die vielen Optionen des Chips nicht alle durchtesten, und testet dann nur die komplizierteste Variante. Und im Errata–Sheet steht natürlich auch nichts.

Inzwischen ist das mecrisp–stellaris bei der Version 2.0.0 angelangt, und der hier beschriebene Ethernet–Treiber wird mitgeliefert.

Senden und Empfangen

Moderne Ethernet–Controller (MAC) arbeiten mit DMA und können mehrere Teile eines Pakets verstreut über den Speicher einlesen. Das ist sinnvoll, weil man es mit Headern und den eigentlichen Daten zu tun hat. Beim Empfangen wäre für das Aufteilen noch ein programmierbarer Filter sinnvoll, damit Pakete verschiedener Protokolle in unterschiedliche Teile zerlegt werden, und am besten gleich direkt im Speicher der Anwendung landen… aber das geht noch nicht. Der Controller kann aber für die üblichen IP–Protokolle die Checksummen berechnen, wenn diese Felder vorher auf 0 gesetzt werden.

Deshalb gibt es als API zum Senden und Empfangen bei mir ein Senden mit einem Block und ein Senden mit zwei Blöcken (Header und Daten), aber kein Empfangen mit zwei Blöcken.

tx-buffer+ *(addr u —)* Reihe ein Paket in den Sendepuffer ein

tx-buffer+2 *(addrhdr uhdr addrdata udata —)* Reihe ein Paket, bestehend aus Header und Daten, in den Sendepuffer ein

rx-buffer+ *(addr u –)* Reihe den Platz für ein Paket in den Empfangspuffer ein

Wenn man ein Paket empfängt, bekommt man einen Descriptor. Der enthält verschiedene Informationen, einen Zeitstempel, und, besonders wichtig, die Adresse und Länge des empfangenen Pakets:

desc@ *(desc – addr u)* Lese die zum Descriptor zugehörige Adresse und Länge aus

Das Empfangen der Pakete erledigt ein Task im Hintergrund. Dafür habe ich einen kleinen kooperativen Multitasker in mecrsip–stellaris eingebaut, für den Matthias Koch die nötigen Primitives geschrieben hat. Mit den Primitives (also SP@, RP@, SP! und RP!) werden dann auch CATCH und THROW implementiert. An oberster Stelle steht also ein Task, der darauf wartet, dass ein Paket ankommt:

```
: ether-loop ( -- )
    BEGIN
        BEGIN  pause
            RX-Descriptor' @ own and 0=
        UNTIL
        RX-Descriptor' dup >r handle-rx
        r> 8 + @ ether-size rx-buffer+
        rx-tail+
    AGAIN ;

: ethernet& ( -- )
    ethernet-task activate
    BEGIN  ['] ether-loop catch  drop
    AGAIN ;
```

Ethernet

Die Pakete auf dem aktuellen Internet haben alle so eine Matroschka–Form, wenn man ein Paket aufmacht, ist erst mal ein kleineres drin, und dann noch eines. Das äußerste Paket ist das Ethernet–Paket. Das sieht (ohne VLAN–Tag) wie in Abbildung 1 aus [2] (die Wikipedia–Artikel sind lesbarer als die eigentlich maßgeblichen RFCs):

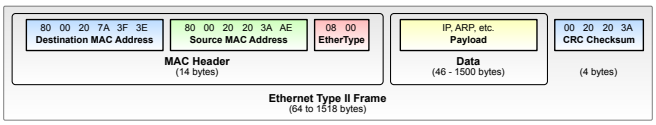


Abbildung 1: Ethernet–Paket

Die Checksumme wird vom MAC schon ausgewertet, kaputte Pakete werden verworfen. Auch beim Senden wird die Checksumme vom MAC berechnet und angehängt. Die passende Zieladresse wird auch vom MAC erledigt (der fühlt sich hier und bei der Broadcast–Adresse — alles FF — angesprochen). Also müssen wir nur noch den EtherType auswerten, der bestimmt, welche Protokolle wir verarbeiten. Interessant für uns sind ARP ($0806), IP ($0800) und IPv6 ($86DD), wobei ich IPv6 noch nicht implementiert habe.

```
' dump-rx 0
' dump-rx 0
' dump-rx 0
' dump-rx 0
' dump-rx 0
10 nvariable free-ethertypes

' rx-arp  $0806
' rx-ipv6 $86DD
' rx-ip   $0800
6 nvariable ethertypes

: dispatcher ( addr u type table -- )
    16 cells bounds do
        dup i @ = if
            drop  i cell+ @ execute
            unloop  exit
        then
    2 cells  +loop  drop 2drop ;

: handle-rx ( desc -- )  desc@
    over eth-type be-w@ ethertypes dispatcher ;
```

NVARIABLE ist ein mecrisp–Konstrukt, mit dem man mehrere Werte als initialisiertes Feld ins Flash schreiben kann — sie werden dann beim Start ins RAM verschoben, damit man sie auch ändern kann. An die niedrigste Adresse landet das, was oben am Stack lag, und auch das Allozieren fängt bei hohen Adressen an.

Address Resolution Protocol

Das erste Protokoll, das man implementieren muss, ist ARP, das Address Resolution Protocol [3]. Bei ARP geht es darum, welche IP–Adresse zu welcher Ethernet–Adresse gehört. Man schickt also einen Broadcast mit „Wer hat diese IP–Adresse“ an alle Ethernet–Knoten im LAN. Und der Empfänger, der sich zuständig fühlt, antwortet mit „Ich habe diese IP–Adresse“. Aus mir nicht ganz verständlichen Gründen enthält ein ARP–Paket noch einige redundante Teile, nämlich die Ethernet–MAC–Adressen, die ja schon im Ethernet–Header stehen. Offenbar hat das Protokoll jemand entworfen, dessen IP–Stack diese wertvollen Informationen schon verschluckt hat, bevor er das Paket bekommt.

Na, egal. Für ARP brauchen wir zunächst mal einen Cache, damit wir die Informationen auch benutzen können, um Pakete nach draußen zu schicken. Ein Cache–Entry enthält also eine IP–Adresse und eine Ethernet–MAC:

```
10 constant /arp
16 constant arp-cache#
/arp arp-cache# * buffer: arp-cache
Variable arp#
: arp-cache+ ( addr u -- )
    arp-cache arp# @ + swap move
    /arp arp# @ +
    dup /arp arp-cache# * u< and arp# ! ;
: ip>mac ( ip-addr -- macaddr/0 )  @
    arp-cache /arp arp-cache# * bounds DO
        dup i 6 + @ = IF
            drop I  UNLOOP  EXIT  THEN
    /arp +LOOP  drop 0 ;
```

Der Cache ist hier einfach ein Ringpuffer, und überschreibt alte Werte, hat aber für 16 Einträge Platz. Solange nicht jeder mit jedem ständig wild durcheinander redet, reicht das völlig aus — und wenn man Probleme hat, ändert man eben die Konstante.

Für die eigentliche Abarbeitung von ARP–Paketen muss ich nur entweder Antwort–Pakete auf Anfragen generieren, oder Antworten in den Cache einsortieren. Beim Zurücksenden mache ich aus einem Request eine Antwort, indem ich die Absender–Angaben in den Empfänger hineinkopiere.

```
: >reply ( addr u -- addr u )
    over dup 6 + swap 6 move ;

: rx-arp ( addr u -- )
    over ether-header# + >r
    r@ 2@ $01000406 $00080100 d=
            \ is it an arp request?
    IF  myip @ r@ 24 + @ =
            \ is it actually for our IP?
        IF  \ arp request: do in-place reply
            >reply  r> 7 + 2 tc,
            \ set reply flag
            dup dup 10 + 10 move
            \ move request to reply tuple
            mymac 6 t$, myip 4 t$,
            \ set my mac
            2drop 42 tx-buffer+ EXIT
            \ reply it
```

```
        THEN
        rdrop 2drop EXIT \ not my IP
    THEN
    r@ 2@ $02000406 $00080100 d=
            \ is it an arp reply?
    IF
        2drop r> 8 + 10 arp-cache+  EXIT
    THEN
    ." ARP unknown packet received "
    r@ 2@ hex. hex. cr
    rdrop .packet cr ;
```

Für Anfragen und „Gratuituos ARP“, also ein ARP–Broadcast, bei dem man unaufgefordert allen mitteilt, wer man ist (das ist primär zum Verstopfen von Caches sinnvoll), gibt es noch zwei Wörter:

grat–arp *(–)* Sende Gratuituos ARP an alle

req–arp *(ip-addr –)* Frage nach dieser IP–Adresse (Antwort taucht dann im Cache auf)

Internet Protocol

Die darauf aufbauenden Protokolle haben einen gemeinsamen Header, den IP–Header [4], siehe Abbildung 2 — außer ARP findet alles im Rahmen von IP statt.

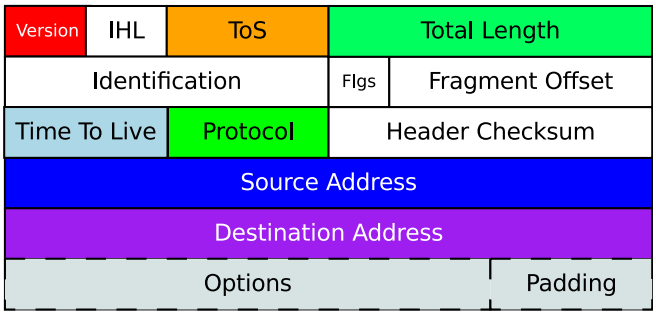


Abbildung 2: IP–Header

Das, was uns beim Empfangen eines IP–Paketes als erstes interessiert, ist das Protokoll. Man müsste vielleicht gucken, ob man mit Destination Address wirklich gemeint ist, aber das kann man auch bei ARP — und sich dann darauf verlassen, dass schon nur ankommt, was man auch bestellt hat. Man könnte auch nach fragmentierten Paketen gucken, aber üblicherweise setzt man heute einfach das „don’t fragment“ Bit, und hält sich an die MTU. Bei Ethernet ist dies 1500 Bytes, durch ein PPPoE hindurch wird es auf 1492 Bytes verkürzt (der zusätzliche PPP-Header kostet 8 Bytes).

```
' .ippacket 0
' .ippacket 0
' .ippacket 0
' .ippacket 0
' .ippacket 0
' .ippacket 0
12 nvariable free-iptypes

' udp-rx 17
' icmp-rx 1
4 nvariable iptypes
: rx-ip ( addr u -- )
    over ip-protocol c@ iptypes dispatcher ;
```

Das Auswerten des Protokolls mache ich wie oben beim Ethernet. Zwei Protokolle werden schon verstanden: ICMP und UDP. Beide Fälle sind zur Zeit nur auf Antworten ausgelegt, müssen also von einem Host erst mal gefragt werden, bevor sie antworten können. Das ist eine für das derzeitige Ziel völlig ausreichende Fähigkeit. Insbesondere braucht man dafür ein Wort, das einen Antwort–Header generiert:

```
: ip-reply ( src dest -- src dest' )
    over eth-src over eth-dest 6 move
    eth-src ffmac, \ will be replaced
    $800 tw,       \ type: IPv4
    over ip-version over 10 move 10 +
    0 tw,          \ checksum 0
    myip over 4 move 4 +
    over ip-src  over 4 move 4 + ;
```

Internet Control Message Protocol

Das ICMP [5] kann eigentlich noch mehr Dinge, wirklich brauchen wir zunächst aber nur Ping–Replies. Also: Wenn wir einen Request vom Typ 8 bekommen haben (also ein Echo Request), dann antworten wir dem mit den gleichen Daten als ICMP Typ 0:

```
: icmp-rx ( addr len -- )
    over icmp-type c@ 8 = IF
        icmp-header# - 4 - >r
        TX-Puffer ip-reply
        0 tl, \ type 0, code 0, cs not set
        swap icmp-header# + swap r@ move
        \ copy rest
        TX-Puffer r> icmp-header# + tx-buffer+
    ELSE
        singletask
        ." ICMP received" cr
        .ippacket
    THEN ;
```

Das war’s schon. Alle anderen ICMP–Typen dumpe ich vorerst mal auf die Ausgabe. Für die passiv lauschende Rolle des Stacks reicht das völlig.

User Datagram Protocol

Hier wird’s erst richtig interessant, weil das unser erstes Protokoll ist, das wirklich von User–Programmen am PC direkt gesprochen werden kann. Matthias Koch schreibt seine Programme am PC in Free Pascal, ich meine in Gforth.

UDP ist ein sehr leichtgewichtiges Protokoll, es belegt lediglich 8 Bytes für seinen Header: Quell–Port, Ziel–Port, Länge und Checksumme. Obwohl es so schlank ist, ist die zweite Hälfte schon überflüssig: Die Länge haben wir jetzt dreimal: Einmal vom Ethernet–Frame, dann vom IP–Header, und jetzt auch noch vom UDP–Header. Welche Länge gilt? Schwer zu sagen: Eigentlich die kleinste

der drei. Aber es gibt da UDP lite, da wird die UDP–Länge ignoriert. Sinn von UDP lite soll sein, dass man Echtzeitanwendungen ohne Verwerfen leicht verfälschter Pakete machen kann, also die Checksumme nicht über die ganzen Daten ausgedehnt wird. Da aber die Checksumme des Ethernet–Frameworks (gilt auch für andere Layer–2–Protokolle) bereits den Transport abgesichert hat, hilft das gar nichts. Es reduziert nur den Aufwand zum Berechnen der Checksumme.

Jedenfalls brauchen wir mal wieder einen Dispatcher, nämlich für jeden Ziel–Port eine Antwort–Routine. Der funktioniert wie oben:

```
' .udppacket 0
' .udppacket 0
' .udppacket 0
' .udppacket 0
' .udppacket 0
' .udppacket 0
12 nvariable free-udpports

' udp-data 4202
' udp-term 4201
4 nvariable udpports

\ udp port dispatcher

: udp-rx ( addr u -- )
    over udp-dest be-w@
    udpports dispatcher ;
```

Zwei Ports sind schon mal vorbelegt: Der eine für Datenlieferungen, der andere für ein Terminal. Das Terminal möchte ich hier nicht in voller Länge zeigen, die Daten–Auslieferung schon. Denn alles, was eine Anfrage auf diesen Port macht, ist, den Header für eine Antwort freischalten. Mit diesem Header kann das Gerät dann jederzeit Daten an den Host senden. Was brauchen wir? Zunächst mal Platz für die Header (das Terminal funktioniert vom Prinzip genauso):

```
\ udp

udp-header# aligned buffer: term-hdr
udp-header# aligned buffer: data-hdr
```

Dann ein Wort, das uns einen solchen Reply–Header aus dem Anfrage–Header erzeugt — vergleiche oben ICMP Ping Reply:

```
: udp-reply ( addr destaddr -- addr destaddr' )
    ip-reply
    over udp-dest be-w@ tw, \ swap dest/src
    over udp-src  be-w@ tw,
    0 tl, ; \ length+CS stub
```

Und dann noch ein Wort, das mit Hilfe so eines Headers eine Antwort zurückschickt. Hier müssen nur zwei Werte im Header ausgetauscht werden: Die Längen für IP und UDP (jeweils Datenlänge mit Header). Diese Information ist redundant.

```
: sendv ( addr u hdr -- ) >r
    dup 8 + r@ udp-len be-w!
    dup 28 + r@ ip-len be-w!
    r> udp-header# 2swap tx-buffer+2 ;
```

Tja, und was macht dann `udp-data` noch? Genau: So einen Header erzeugen, mehr nicht. Beim Terminal werden die gesendeten Daten noch als Tastendrucke in einen Puffer geschrieben, und weitergereicht.

```
: udp-data ( addr u -- )
    \ just setup the reply buffer
    drop data-hdr udp-reply 2drop ;
```

Matthias Koch hat aufgrund dieses Protokolls ein kleines Scope geschrieben, in der Datei `ether-adc.txt`, welches die Daten vom ADC in Kilobyte–Blöcken an den Host schickt. Der ADC sampelt fleißig in einen 2kB großen Ringpuffer.

```
: capture ( Samples -- )
  0 sampleno! samples!
  \ Start at the beginning of the buffer
  begin
    begin pause sampleno@ 512 u>= until
    \ Wait for the sample number to be
    \ in the second half of the buffer
    Circus 1024 data-hdr sendv
    \ Transmit first half of the buffer
    begin pause sampleno@ 512 u< until
    \ Wait for the sample number to be
    \ in the first half of the buffer
    Circus 1024 + 1024 data-hdr sendv
    \ Transmit second half of the buffer
  samples@ 0= until
  \ Continue until all samples are transmitted.
;
```

Referenzen

[1] Matthias Koch, *Mecrisp Stellaris,* `http://mecrisp.sourceforge.net/`

[2] Wikipedia, *Ethernet Frame,* `https://en.wikipedia.org/wiki/Ethernet_frame`

[3] Wikipedia, *Address Resolution Protocol,* `https://de.wikipedia.org/wiki/Address_Resolution_Protocol`

[4] Wikipedia, *IPv4,* `https://de.wikipedia.org/wiki/IPv4`

[5] Wikipedia, *ICMP,* `https://de.wikipedia.org/wiki/Internet_Control_Message_Protocol`

[6] Wikipedia, *UDP,* `https://de.wikipedia.org/wiki/User_Datagram_Protocol`

Protokoll der Mitgliederversammlung 2014

Carsten Strotmann

Ort:
College Garden Hotel
Johann–Strauß–Straße 2
2540 Bad Vöslau / Österreich

<2014-03-30 Sun 09:31> Beginn der Mitgliederversammlung
<2014-03-30 Sun 09:33> Schriftführer Carsten Strotmann
<2014-03-30 Sun 09:34> Versammlungsleiter Heinz Schnitter
<2014-03-30 Sun 09:37> Heinz Schnitter übernimmt die Versammlungsleitung
<2014-03-30 Sun 09:37> Beschlussfähigkeit gegeben: 16 Mitglieder Mitgliederstand Jahresende 2013: 108 Mitglieder
<2014-03-30 Sun 09:39> Ewald Rieger - Bericht des Vorstands Mitgliederentwicklung weiter leicht rückläufig

2	Eintritte 2013
5	Austritte 2013
2	Eintritte 2014

<2014-03-30 Sun 09:44> Der Einzug der Mitgliedsbeiträge wurde Ende 2013 auf das neue SEPA Verfahren umgestellt
<2014-03-30 Sun 09:49> Ein/Ausgaben

Posten	Einnahmen	Ausgaben
Ideeller Bereich	3396,33	2387,57
Vermögensverwaltung		136,58
Zweckbetrieb	2079,95	3042,48

Verlust: 90,35 Euro

<2014-03-30 Sun 09:59> Wirtschaftsplan 2014

Einnahmen:	3800 Euro
Ausgaben:	6520 Euro
Verlust:	ca. 2700 Euro geplant

<2014-03-30 Sun 10:02> Thomas Prinz hat die Kasse geprüft und keine Unstimmigkeiten festgestellt. Über die Entlastung des Kassierers wird abgestimmt: 14 Stimmen für eine Entlastung, 0 Gegenstimmen, 3 Enthaltungen

<2014-03-30 Sun 10:04> rund um das Forth-Magazin (VD) Die Editoren für die Vereinszeitschrift "Vierte Dimension" im Jahr 2014 sind: 02/14 Carsten Strotmann, 03/14 Ulrich Hoffmann, 04/14 Bernd Paysan

Michael Kalus und Matthias Koch planen ein ARM-Sonderheft.

<2014-03-30 Sun 10:15> Internet Präsenz: Bernd Paysan und Ulrich Hoffmann arbeiten an der Auswahl eines neuen Blog-Systems für die Web-Präsenz der Forth-Gesellschaft

<2014-03-30 Sun 10:25> Bericht über die Außendarstellung: Die Forth-Gesellschaft war im vergangenen Jahr auf folgenden Veranstaltungen Präsent: LinuxTag 2013, MakerFaire 2013, 30C3 Hamburg

Für das Jahr 2014 sind Beteiligungen an weiteren Veranstaltungen geplant:

VCFe 2014	3.-4. Mai	`http://vcfe.org`
FrosCon St. Augustin	24.-24. August	`http://froscon.de`
Vintage Computer Festival Berlin	3.-5. Oktober	`http://vcfb.de`
OpenRheinRuhr	8.-9. November	`http://openrheinruhr.de`
31C3 Hamburg	27.-30. Dezember	

<2014-03-30 Sun 10:41> Abstimmung über die Entlastung des Direktoriums: 15 Stimmen für die Entlastung, 2 Enthaltungen, 0 Gegenstimmen

<2014-03-30 Sun 10:42> Wahl des Direktoriums: Die Mitglieder des aktuellen Direktoriums stellen sich wieder zur Wahl (Ewald Rieger, Bernd Paysan, Ulrich Hoffmann): 16 Stimmen für eine Wiederwahl, 1 Enthaltung, 0 Gegenstimmen

<2014-03-30 Sun 10:44> Verleihung des Drachenpreises: Willi Stricker ist der Preisträger des Drachenpreises 2014

<2014-03-30 Sun 10:49> Fototermin
<2014-03-30 Sun 11:13> Fortsetzung der Tagung

Diskussion: Verschiedenes

Gruppe "Ruhrpott"

Martin Bitter, Michael Kalus und Carsten Strotmann organisieren eine neue Forth-Regionalgruppe im Ruhrgebiet. Treffpunkt ist monatlich zu wechselnden Terminen das Unperfekthaus in der Innenstadt von Essen: (`http://www.unperfekthaus.de`). Das aktuelle Projekt der Gruppe ist eine Anpassung von gForth an das LEGO-Mindstorms-EV3-System. Die Gruppe hat hierzu eine finanzielle Unterstützung in Höhe von ca. 300 Euro beantragt, über welche das Direktorium entscheiden wird. Termine für die Treffen der Gruppe "Ruhrpott" finden sich auf der Webseite der Forth-Gesellschaft (`http://forth-ev.de` oder können bei den oben genannten Organisatoren per E-Mail erfragt werden.

Projekt: Forth-IDE

Angeregt wurde die Erstellung einer Forth-IDE (Integrated Development Environment) mit grafischer Oberfläche. Gerd Franzkowiak bot an, eine Python-basierte IDE dem Projekt zur Verfügung zu stellen. Informationen im Projekt-Verzeichnis im Wiki der Forth-Gesellschaft-Webseite.

Internet - theforth.net

Gerald Wodni kündigte an, dass die Webseite `http://theforth.net` in der Zukunft auch eine Schnittstelle für das Quellcode-Verwaltungssystem "git" zur Verfügung stellen wird.

Projekt: Modell-Forth

Das Projekt "Modell-Forth" hat das Ziel, ein neues, modernes Modell-Forth im Quellcode zu erstellen. Das Modell-Forth soll es Programmierern ermöglichen, die Funktionsweise einer Forth-VM zu verstehen und anzupassen. Es wird daher im ersten Schritt auf eine Forth-Meta-Compilation verzichtet und das Modell-Forth als Assembler-Listing aufgebaut. Ein Forth-Meta-Compiler, welcher Assembler-Code für Forth-Definitionen erzeugt, ist als Option geplant. Dokumentiert wird das neue Modell-Forth über ein "Systemguide", wie die frühen FIG-Forth- und Forth83-Versionen.

Das Modell-Forth soll dem neusten Forth-200x-Standard entsprechen, jedoch minimal und einfach gehalten sein. Der Quellcode wird unter einer liberalen OpenSource-Lizenz stehen.

Als Ausgangspunkt wird Mecrisp (Matthias Koch), ciforth (Albert v.d. Horst), no-forth (Albert Nijhof & Willem Ouwerkerk) und gforth-ec genommen.

Koordiniert wird dieses Projekt durch Ulrich Hoffmann und Carsten Strotmann.

CCC-Kontakte

Es wurde angeregt, Kontakte zu lokalen Gruppen des Chaos-Computer-Clubs (CCC) aufzubauen. Viele Aktive im CCC beschäftigen sich mit der Mikrocontroller-Programmierung, so dass ein Erfahrungsaustausch CCC-Mitglieder an Forth interessieren kann.

- Bernd Paysan/Heinz Schnitter - München
- Martin Bitter/Carsten Strotmann - Essen
- Erich Wälde - Freiburg
- Gerd Franzkowiak - Ulm

Vintage Computer Festival Europe

Heinz Schnitter wird auf dem Vintage Computer Festival im Mai in München seinen Vortrag über das Wickeln der Spulen für den Beschleuniger in Garching anbieten.

Carsten Strotmann wird wie in den vergangenen Jahren den Forth-Benchmark-Wettbewerb auf dem VCFe betreuen.

Forth-Tagung 2015 in Hannover

Die Tagung der Forth-Gesellschaft im Jahre 2015 wird in Hannover stattfinden. Der Termin wird um/in den Osterferien 2015 liegen, der genaue Termin wird in der Ausgabe 03-2014 der VD bekanntgegeben. Organisiert wird die Tagung von Bernd Paysan, Carsten Strotmann und Willi Stricker.

Projekt: OpenNetworkForth (aka ONF)

Das "Internet-der-Dinge" (Vernetzte Mikrocontroller) ist derzeit ein populäres Thema in der Informationstechnik. Hierfür soll das von Heinz Schnitter in den 1980er Jahren entwickelte OpenNetworkForth (ONF) auf moderne Forth-Systeme und Entwicklungsboards angepasst werden. Für diesen Network-Stack in Forth für verteilte Systeme wird eine Modell-Implementierung erarbeitet und dokumentiert.

Organisiert wird dieses Projekt von Heinz Schnitter, Bernd Paysan, Gerald Wodni und Egmont Woizel.

Informationen und Dokumentation zu diesem Projekt findet sich im Projekt-Verzeichnis des Wiki auf der Webseite der Forth-Gesellschaft.

<2014-03-30 Sun 12:17> Ende der Mitgliederversammlung

Die Vortragsvideos und Folien der Vorträge der Tagung 2014 sind unter `http://www.forth-ev.de/wiki/doku.php/events:tagung-2014` abrufbar.

Abbildung 1: Gruppenfoto von der Tagung

Smartfirmware unter Linux kompilieren

Carsten Strotmann

Smartfirmware ist eine Implementation des OpenFirmware(IEEE 1275–1994)–Standards. Im Jahr 2005 hat CodeGen, die Firma hinter Smartfirmware, den Quellcode unter einer freien BSD-ähnlichen Lizenz veröffentlicht.

Smartfirmware

Die freien OpenFirmware–Implementationen (Links siehe unten) sind eine Fundgrube für Forth–Programmierer, hier finden sich viele hilfreiche Programmteile (TCP/IP für IPv4 und IPv6, Ethernet-Treiber, USB, FAT- und Unix-Dateisystemtreiber, Kryptographie-Routinen) unter freien OpenSource–Lizenzen.

Der Smartfirmware–Quellcode beinhaltet als Besonderheit einen C–zu–FCode(Forth)–Compiler, mit dem C–Programme in Forth–Programme übersetzt werden können.

Der Quellcode im Repository unter `openfirmware.info` lässt sich nicht mit einem modernen C-Compiler unter Linux übersetzen. Ich habe daher das Quellpaket unter "github" importiert und den C-Quellcode für Linux angepasst.

Smartfirmware bauen

Zum Bauen von Smartfirmware braucht man einen C-Compiler (GNU C-Compiler "gcc" oder LLVM "clang") sowie das Softwareverwaltungsprogramm "git":

1. Den Quellcode auschecken:
   ```
   % git clone https://github.com/cstrotm/smartfirmware.git
   ```
2. Den Basis–Quellcode übersetzen. Der Quellcode ist nicht 64Bit "clean", so dass die Übersetzung derzeit nur auf einem 32Bit–Linux funktioniert:
   ```
   % cd smartfirmware
   % sudo ./makeworld.sh
   ```
3. Nicht wundern, das `makeworld.sh`–Script legt ein Verzeichnis `/cgt` im Hauptverzeichnis des Linux–Dateisystems an. In diesem Verzeichnis befinden sich die Smartfirmware–Programme.
4. Nun kann der Smartfirmware–Interpreter für Unix übersetzt werden:
   ```
   % cd bin/of/unix
   % make
   ```
5. Der Smartfirmware–Interpreter ist nun gebaut und kann ausprobiert werden:
   ```
   % ./of
   Welcome to SmartFirmware(tm) for CodeGenInc Unix version 1.00
   SmartFirmware(tm) Copyright 1996-2001 by CodeGen, Inc.
   All Rights Reserved.
   Extended diagnostics are now switched on.
   ok
   ```

Listings

C-Code

```
1  //#include <stdio.h>
2  extern int printf(char const *fmt, ...);
3
4  int
5  main(int, char **)
6  {
7      printf("Hello, world!\n");
8      return 0;
9  }
```

FCode Compiler

```
1  % ../cc-fcode hello.c
2  fcode-version2
3  headerless
4
5  0 value $frame \ C stack-frame pointer
6
7  variable main$!unnamed-1
8  variable main$!unnamed-2
9
10 \ main
11  main$!unnamed-2 l!
12  main$!unnamed-1 l!
13  \ saved parameters
14  " Hello, world!"(0A)"(00)" drop printf drop
15  0 exit
16 \ end main
17 headerless
18
19
20 fcode-end
```

Literatur und Links

1. Smartfirmware Repository on Github `https://github.com/cstrotm/smartfirmware`
2. OpenFirmware Quellcode `http://www.openfirmware.info/Open_Firmware`
3. Smartfirmware `http://www.openfirmware.info/SmartFirmware`
4. SUN OpenBoot `http://www.openfirmware.info/OpenBOOT`
5. OpenBIOS `http://www.openfirmware.info/OpenBIOS`
6. Slimline Open Firmware `http://www.openfirmware.info/SLOF`

Forth–Gruppen regional

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

München **Bernd Paysan**
Tel.: (0 89) – 41 15 46 53 (p)
`bernd.paysan@gmx.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00 in der Pizzeria La Capannina, Weitlstr. 142, 80995 München (Feldmochinger Anger).

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

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

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

Gruppengründungen, Kontakte

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

μP–Controller Verleih

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

Spezielle Fachgebiete

Forth-Hardware in VHDL microcore (uCore)
Klaus Schleisiek
Tel.: **(0 75 45) – 94 97 59 3** (p)
kschleisiek@freenet.de

KI, Object Oriented Forth, Sicherheitskritische Systeme
Ulrich Hoffmann
Tel.: **(0 43 51)** – **71 22 17** (p)
Fax: – **71 22 16**

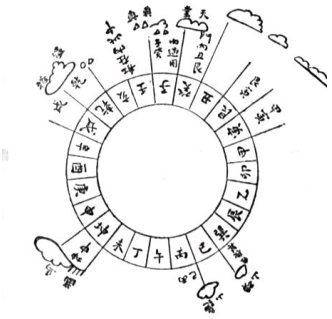
Forth–Vertrieb
volksFORTH
ultraFORTH
RTX / FG / Super8
KK–FORTH
Ingenieurbüro
Klaus Kohl–Schöpe
Tel.: **(0 82 66)–36 09 862** (p)

Termine

Donnerstags ab 20:00 Uhr
Forth–Chat IRC `#forth-ev`

23.–24. August 2014:
FrOSCon Hochschule Bonn-Rhein-Sieg
`http://froscon.de`
29. August 2014: Treffen der Forth-Gruppe "Ruhrpott"
3.–5. Oktober 2014:
Vintage Computing Festival Berlin (VCFB)
`http://vcfb.de`
8.–9. November 2014:
OpenRheinRuhr `http://openrheinruhr.de`
27.-30. Dezember 2014:
31C3 Chaos Communication Congress Hamburg
`http://ccc.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

Euro–Forth–Konferenz 2014 vom 23. bis 29. September in Palma de Mallorca

 [1]

 [2]

Unterbringung und Tagung im Hotel Amic Horizonte (`http://hotelamichorizonte.com`). Die Organisation der EuroForth 2014 wird von Nick und Janet Nelson durchgeführt.

Homepage und weitere Informationen

Die Euro-Forth 2014 Homepage unter `http://www.complang.tuwien.ac.at/anton/euroforth/ef14/` bietet weitere Informationen und das aktuelle Konferenzprogramm.

Anmeldung

Im Informationstext `http://www.euroforth.org/ef14/announcement.pdf` finden sich die Anmeldeinformationen.

Programm

24.-26. September

Forth200x meeting

26.-28. September

Euro-Forth 2014 Konferenz

29. September

Optionaler Tag

[1] Quelle: wikipedia, cc-by-sa, Cathedral of Palma de Mallorca, Uploaded by StAn

[2] Quelle: wikipedia, cc-by-sa, Harbour of Palma with the Castle of Bellver in the background by Hedwig Storch